

ENKBB-2.0

VAX 11/730 8085
TEST PART II

EP-ENKBB-DL-2.0
FICHE 1 OF 1

JAN 1983
COPYRIGHT© 82-83
MADE IN USA



IDENTIFICATION

Product code: ENKBB VERSION 2.0
Product name: 8085 BASED MICRO-DIAGNOSTIC FOR VAX-11/730 WCS
AND CPU MODULES
Product date: 11-OCT-82
Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC
DECUS
UNIBUS

DECsystem-10
MASSBUS
VAX

DECSYSTEM-20
PDP
VMS

digital

Table of Contents

1.0	ABSTRACT	3
2.0	HARDWARE REQUIREMENTS	3
3.0	SOFTWARE REQUIREMENTS	3
4.0	PREREQUISITES	3
5.0	OPERATING INSTRUCTIONS	3
5.1	Options	3
5.2	Event Flags	4
5.3	APT Setup	4
6.0	PROGRAM FUNCTIONAL DESCRIPTION	4
6.1	Program Overview	4
6.2	Program Size	4
6.3	Program Run Times	4
6.4	Run-time Dynamics	4
6.5	Fault Detection	5
6.6	Performance During Hardware Failures	5
6.7	Program Applications	5
6.8	Test Descriptions	6
7.0	MAINTENANCE HISTORY	6

1.0 ABSTRACT

This program is an 8085 based micro-diagnostic designed to test the hardware on the WCS (Writable Control Store) and DAP (Data Path) modules of the VAX-11/730 processor.

2.0 HARDWARE REQUIREMENTS

This program will run with a minimum hardware configuration of the WCS and CPU modules present.

Other hardware modules and options may be installed without affecting the diagnostic's performance.

3.0 SOFTWARE REQUIREMENTS

This diagnostic must be run under the VAX-11/730 micro monitor (ENKAA) in a standalone environment. The micro-diagnostic is written into the top 3K of 8085 RAM. The micro-diagnostic monitor also resides in 8085 ram, where console software is normally present.

4.0 PREREQUISITES

This module should be run after running the module ENKBA.

5.0 OPERATING INSTRUCTIONS

This program may be executed by typing DI SE ENKBB after the micro-monitor has been loaded and the MIC> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette or downline loaded through the APT-RD port. See the OVERVIEW MANUAL for more information.

5.1 Options

Not applicable

5.2 Event Flags

Not applicable

5.3 APT Setup

The micro-monitor knows when APT is present by the position of the keyswitch and the state of a line connected to the APT-RD port.

Under APT, the diagnostic will halt on error and report error information to APT via the mailbox in 8085 RAM.

The following describes the APT parameters needed to execute this diagnostic:
TBS

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

This program is designed to detect stuck-at faults and provide a repair tool which helps isolate the failing component. A bottom up diagnostic strategy is utilized which builds on previous tests. The tests should be run in consecutive order, to gain the best isolation of a failing component.

6.2 Program Size

This program runs in 8085 RAM memory and is approximately 3K bytes long.

6.3 Program Run Times

This program will take approximately 1 minute and 40 seconds to run.

6.4 Run-time Dynamics

Not applicable

6.5 Fault Detection

The diagnostic will report errors with the following header:

SECT TST ERR EXP REC OTHER MSK MODULE

1. SECT is the program name (ENKBB)
2. TST is the test number that failed.
3. ERR is the error number that failed.
4. EXP is the expected (correct) data.
5. REC is the received (actual) data.
6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the listing will describe the contents of this field when it is used.
7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.
8. MODULE is the suspected module that caused the failure.

6.6 Performance During Hardware Failures

A power fail will cause a rebooting of the system (if a battery backup is not present). Other hardware failures will cause normal error printouts.

6.7 Program Applications

This program can be used by field service to determine which module should be replaced and to verify that the replacement eliminated the failure. It can also be used as a repair tool to help isolate a failing component by manufacturing, field service or engineering. The program is API compatible.

Customers may use the program to verify the basic integrity of the WCS and CPU modules in the system.

6.8 Test Descriptions

See the actual test in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

7.0 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
8-MAR-82	1.0	Initial release.
11-OCT-82	2.0	RD changes.

22-ENKBB-2.0 7000 4
7000 3
7000 4
7000 5
7000 6
7000 7
7000 8
7000 9
7000 10
7000 11
7000 12
7000 13
7000 14
7000 15
7000 16
7000 17
7000 18
7000 19
7000 20
7000 21
7000 22
7000 23
7000 24
7000 25
7000 26
7000 27
7000 28
7000 29
7000 30
7000 31
7000 32
7000 33
7000 34
7000 35
7000 36
7000 37
7000 38
7000 39
7000 40
7000 41
7000 42
7000 43
7000 44
7000 45
7000 46
7000 47
7000 48
7000 49
7000 50
7000 51
7000 52
7000 53
7000 54
7000 55
7000 56
7000 57
7000 58
7000 59
7000 60
7000 61
7000 62

H 1

Fiche 1 Frame H1 Sequence 7

ENKBB 8085 based diagnostic for WCS/DAP module Rev 02.00
Table of Contents

135	DATA AREA
302	PROGRAM ENTRY POINT
322	VARIABLES AREA
471	PROGRAM SPECIFIC SUBROUTINES
472	CON SET FOR_CO
489	CHECK_RD
510	LD_SUPPORT
567	SET_COMPARE
593	LS_WRITE
623	LS_READ
653	FAST_WRITE_WRO
703	FAST_READ_WRO
732	READ_BYTE_WRO
762	WRITE_WR1_32
800	READ_WR1_32
842	SHIFT_WRT_R8
869	EXEC_INST
875	LD_CSR_INST
916	LOAD_MOD_DAT
930	TEST9_1_0
963	TEST9_SHIFT
1001	TEST9_2_0
1047	TEST9_ERROR
1050	TEST9_ERR2_3
1074	TESTA_ERROR
1100	TESTB_ERROR
1121	TESTD_ERROR
1124	TESTC_ERROR
1147	TESTE_F_ERROR
1175	REPORT_CPU_ERROR
1179	REPORT_ERROR
1238	TEST 9 - 2901 DATA PATH TEST (WORKING REG 0 AND WR CRR TESTS)
1398	TEST A - 2901 SHIFT FUNCTION TEST
1569	TEST B - WORKING REGISTER CLEAR TEST
1881	TEST C - WORKING REGISTER TEST (WORKING REG 1)
2152	TEST D - 32 BIT D AND Y BUSES TEST
2399	TEST E - LS RAM DATA TEST
2535	TEST F - LS RAM MARCH PATTERN TEST

7000	63
7000	64
7000	65
7000	66
7000	67
7000	68
7000	69
7000	70
7000	71
7000	72
7000	73
7000	74
7000	75
7000	76
7000	77
7000	78
7000	79
7000	80
7000	81
7000	82
7000	83
7000	84
7000	85
7000	86
7000	87
7000	88
7000	89
7000	90
7000	91
7000	92
7000	93
7000	94
7000	95
7000	96
7000	97
7000	98
7000	99
7000	100
7000	106
7000	107
7000	108
7000	109
7000	110
7000	111
7000	112
7000	113
7000	114
7000	115
7000	116
7000	117
7000	118
7000	119
7000	120
7000	121
7000	122
7000	123
7000	124
7000	125
7000	126
7000	127

```

; MODIFIED BY: David Mayo      8-MAR-1982      VERSION 01.00
;              Ernie Preisig  26-Aug-1982      Version 02.00      RD Changes

```

• • •

```

7000 128
7000 129
7000 130
7000 131
7000 132
7000 133
7000 134
7000 135
7000 136 000
7000 137
7000 138
7000 139
7000 140
7000 141
7000 142
7000 143
7000 144
7000 145 00000000
7000 146
7000 147 00000000
                                00000002
                                00000004
                                00000006
                                00000006
                                00000001
                                00000003
                                00000005
                                00000007 00000006
7000 148
7000 149 00000000
7000 150
7000 151
7000 152 00000001
7000 153 00000001
7000 154
7000 155
7000 156 00000002
7000 157
7000 158 00000002
7000 159
7000 160
7000 161 00000003
7000 162
7000 163 00000003
7000 164
7000 165
7000 166 00000004
7000 167 00000004
7000 168
7000 169 00000005
7000 170
7000 171 00000006
7000 172
7000 173 00000007
7000 174
7000 175 00000007
7000 176
7000 177
7000 178 00000008
7000 179 00000009

                                .PSECT CPU2,BYTE
                                .ALIGN BYTE
                                *****
                                ; DATA AREA
                                *****
HEAD                                =0
                                EQUATE                                ;REGISTER EQUATE MACRO

UPC14A                                =<^X00>                                ;(R0) UPC BIT 14 ASSERTED HIGH <MSB>

MISC2                                =<^X01>
BOOTF                                =<^X01>                                ;(R0) BOOT FLAG. ASSERTED LOW. <LSB>

DCLO                                =<^X02>                                ;(R0) DC LOW FLAG ASSERTED HIGH<MSB>
HLTFLG                                =<^X02>                                ;(R0) HALT FLAG ASSERTED LOW <LSB>

READJ1                                =<^X03>                                ;(R0) JUMPER J1 FLAG ASS. HIGH <MSB>
ALLOWP                                =<^X03>                                ;(R0) ENABLE CONTROL P ASS LOW <LSB>

READJ2                                =<^X04>                                ;(R0) BIT 07 EQUALS J2 <MSB>
APTFLG                                =<^X04>                                ;(R0) APT PRESENT FLAG <LSB>

AUTO                                =<^X05>                                ;(R0) AUTO TEST FLAG ASS LOW <LSB>

REMDIS                                =<^X06>                                ;(R0) REMOTE DISABLE FLAG ASS LOW <LSB>

OKV15                                =<^X07>                                ;(R0) 15 VOLTS OK FLAG ASS LOW <MSB>
BOOTEN                                =<^X07>                                ;(R0) BOOT ENABLE FLAG ASS LOW <LSB>

WCSB0                                =<^X08>                                ;(W0) WCS WRITE DATA REGISTER
WCSB1                                =<^X09>

```

7000 180 0000000A	WCSB2	=<^X0A>	
7000 181			
7000 182 0000001C	ITDB	=<^X1C>	;(R/W) TIMER DATA REGISTER
7000 183 0000001D	ITCSR	=<^X1D>	;(R/W) TIMER STATUS REGISTER
7000 184			
7000 185 00000020	CLRCLK	=<^X20>	;(WO) CLEAR CPU CLOCK RUN
7000 186 00000021	SETCLK	=<^X21>	;(WO) SET CPU CLOCK RUN
7000 187			
7000 188 00000022	CLRSS	=<^X22>	;(WO) CLEAR CLOCK SINGLE STEP
7000 189 00000023	SETSS	=<^X23>	;(WO) SET CLOCK SINGLE STEP
7000 190			
7000 191 00000024	CLRCCK	=<^X24>	;(WO) CLEAR CSR CLOCK
7000 192 00000025	SETCCK	=<^X25>	;(WO) SET CSR CLOCK
7000 193			
7000 194 00000026	CLRUPK	=<^X26>	;(WO) CLEAR THE UPC CLOCK
7000 195 00000027	SETUPK	=<^X27>	;(WO) SET THE UPC CLOCK
7000 196			
7000 197 00000028	SETMCI	=<^X28>	;(WO) SET MEMORY CONTROL INIT
7000 198 00000029	CLRMCI	=<^X29>	;(WO) CLEAR MEMORY CONTROL INIT
7000 199			
7000 200 0000002A	SETMCK	=<^X2A>	;(WO) SET MEMORY CONTROL CLOCK
7000 201 0000002B	CLRMCK	=<^X2B>	;(WO) CLEAR MEMORY CONTROL CLOCK
7000 202			
7000 203 0000002C	CLRTMR	=<^X2C>	;(WO) STOP TIMER
7000 204 0000002D	SETTMR	=<^X2D>	;(WO) START TIMER COUNTING
7000 205			
7000 206 0000002E	CLRBSY	=<^X2E>	;(WO) CLEAR UNIBUS BUS BUSY
7000 207 0000002F	SETBSY	=<^X2F>	;(WO) SET UNIBUS BUS BUSY
7000 208			
7000 209 00000030	SETDCL	=<^X30>	;(WO) SET DC LOW
7000 210 00000031	CLRDCL	=<^X31>	;(WO) CLEAR DC LOW
7000 211			
7000 212 00000032	SETACL	=<^X32>	;(WO) SET AC LOW
7000 213 00000033	CLRACL	=<^X33>	;(WO) CLEAR AC LOW
7000 214			
7000 215 00000034	INITL	=<^X34>	;(WO) CLEAR UNIBUS INIT
7000 216 00000035	INITH	=<^X35>	;(WO) SET UNIBUS INIT
7000 217			
7000 218 00000036	CLRWCK	=<^X36>	;(WO) CLEAR WCS BIT 00
7000 219 00000037	SETWCK	=<^X37>	;(WO) SET WCS BIT 00
7000 220			
7000 221 00000038	CLRCCK	=<^X38>	;(WO) CLEAR WCS BIT 01
7000 222 00000039	SETCCK	=<^X39>	;(WO) SET WCS BIT 01
7000 223			
7000 224 0000003C	CLRLIT	=<^X3C>	;(WO) CLEAR THE RUN LIGHT
7000 225 0000003D	SETLIT	=<^X3D>	;(WO) SET THE RUN LIGHT
7000 226			
7000 227 00000044	TUDB	=<^X44>	;(R/W) TU-58 DATA BUFFER
7000 228 00000045	TUSR	=<^X45>	;(RO) TU-58 STATUS REGISTER
7000 229 00000046	TUMODE	=<^X46>	;(R/W) TU-58 MODE REGISTER 1 + 2
7000 230			;(R/W) TU-58 MODE REGISTER 1 + 2
7000 231			;(R/W) TU-58 MODE REGISTER 1 + 2
7000 232			;(R/W) TU-58 MODE REGISTER 1 + 2
7000 233 00000047	TUCR	=<^X47>	;(R/W) TU-58 COMMAND REGISTER
7000 234			
7000 235 00000048	TTDB	=<^X48>	;(R/W) TERMINAL DATA BUFFER
7000 236 00000049	TTSR	=<^X49>	;(RO) TERMINAL STATUS REGISTER
7000 237 0000004A	TTMODE	=<^X4A>	;(R/W) TERMINAL MODE REGISTER 1 + 2
7000 238			;(R/W) TERMINAL MODE REGISTER 1 + 2
7000 239			;(R/W) TERMINAL MODE REGISTER 1 + 2

7000 240					; POINTER BACK TO MR1
7000 241 0000004B	TTCR	=<^X4B>			; (R/W) TERMINAL COMMAND REGISTER
7000 242					
7000 243 00000080	READ	=<^X80>			; (RO) CONSOLE READ REGISTER
7000 244					
7000 245 00000082	CPUACK	=<^X82>			; (RO) ACKNOWLEDGE FROM CPU <LSB>
7000 246					
7000 247 00000083	CPATTN	=<^X83>			; (RO) ATTENTION FROM CPU <LSB>
7000 248					
7000 249 00000084	UPC14	=<^X84>			; (RO) UPC BIT 14 <LSB>
7000 250					
7000 251 00000085	CSR07	=<^X85>			; (RO) CSR BIT 7 <LSB>
7000 252					
7000 253 00000086	CSR15	=<^X86>			; (RO) CSR BIT 15 <LSB>
7000 254					
7000 255 00000087	CSR23	=<^X87>			; (RO) CSR BIT 23 <LSB>
7000 256					
7000 257 000000A0	ENBPAR	=<^XA0>			; (WO) ENABLE STALL ON PARITY ERROR
7000 258 000000A1	DISPAR	=<^XA1>			; (WO) DISABLE STALL ON PARITY ERROR
7000 259					
7000 260 000000A2	VSHIFT	=<^XA2>			; (WO) SET THE V-BUS TO SHIFT MODE
7000 261 000000A3	VNORML	=<^XA3>			; (WO) SET THE V-BUS TO NORMAL MODE
7000 262					
7000 263 000000A4	SETTIM	=<^XA4>			; (WO) SET THE INTERVAL TIMER INTERRUPT
7000 264 000000A5	CLRTIM	=<^XA5>			; (WO) CLEAR THE INTERVAL TIMER INTRPT.
7000 265					
7000 266 000000A6	ENBMEM	=<^XA6>			; (WO) ENABLE MEMORY REFERENCES
7000 267 000000A7	DISMEM	=<^XA7>			; (WO) DISABLE MEMORY REFERENCES
7000 268					
7000 269 000000A8	SETACK	=<^XA8>			; (WO) SET CONSOLE ACKNOWLEDGE
7000 270 000000A9	CLRACK	=<^XA9>			; (WO) CLEAR CONSOLE ACKNOWLEDGE
7000 271					
7000 272					
7000 273					
7000 274					
7000 275					
7000 276 000000A8	CLRUPC	=<^XA8>			; (WO) CLEAR THE V-BUS SERIAL INPUT
7000 277 000000A9	SETUPC	=<^XA9>			; (WO) SET THE V-BUS SERIAL INPUT
7000 278					
7000 279 000000AA	SETATN	=<^XAA>			; (WO) SET CONSOLE ATTENTION
7000 280 000000AB	CLRATN	=<^XAB>			; (WO) CLEAR CONSOLE ATTENTION
7000 281					
7000 282 000000AC	SETPFI	=<^XAC>			; (WO) SET POWER FAIL INTERRUPT
7000 283 000000AD	CLRPFI	=<^XAD>			; (WO) CLEAR POWER FAIL INTERRUPT
7000 284					
7000 285 000000AE	SETHLT	=<^XAE>			; (WO) SET THE CONSOLE HALT BIT
7000 286 000000AF	CLRHLT	=<^XAF>			; (WO) CLEAR THE CONSOLE HALT BIT
7000 287					
7000 288 000000CC	WRITE	=<^XCC>			; (WO) THE CONSOLE WRITE REGISTER
7000 289					
7000 290 000000EC	YBUSRD	=<^XEC>			; (RO) Y-BUS READ
7000 291					
7000 292					
7000 293					
7000 294					
7000 295					
7000 296 00004C22	PARITY VECTOR	= <^X4C22>			; PARITY ERROR BRANCH VECTOR.
7000 297 00004C25	POWER VECTOR	= <^X4C25>			; POWER FAIL BRANCH VECTOR.
7000 298 000040D9	MIPFLG	=<^X40D9>			; ADDRESS OF MODEM IN PROGRESS FLAG
7000 299					

;ADDITIONAL DATA

```

22-ENKBB-2.0      7000      4
7000 300
7000 301
7000 302
7000 303
7000 304
7000 305
7000 306
7000 307 0450'C3
7003 308
7003 309
7003 310
7003 311
7003 312
7003 313
7003 314
7003 315
7003 316 0200
7005 317 42 42
7007 318
7007 319
7007 320
7007 321
7007 322
7007 323
7007 324
7007 325
7007 326 00000008
7008 327 04
7009 328 00
700A 329 0000
700C 330 00
700D 331 00
700E 332
700E 333 37
700F 334 F8
7010 335 15
7011 336
7011 337 BE
7012 338 00
7013 339 15
7014 340
7014 341 36
7015 342 00
7016 343 15
7017 344
7017 345 BE
7018 346 00
7019 347 15
701A 348
701A 349 36
701B 350 00
701C 351 15
701D 352
701D 353 A0
701E 354 02
701F 355 95
7020 356
7020 357 A3
7021 358 40
7022 359 95

```

M 1

Fiche 1 Frame M1

Sequence 12

```

*****
: PROGRAM ENTRY POINT.... MUST BE THE FIRST INSTUCTION IN THE PROGRAM
*****
ENTRY:          JMP      TEST9

*****
: VERSION NUMBER AND LAST TWO LETTERS OF FILE NAME.  THESE 2 WORDS CAN NOT
: BE MOVED TO ANY OTHER LOCATION WITHOUT A MICMON CHANGE.
*****
VERSION:         .WORD      <^X0200>          ;VERSION NUMBER (REV 02.00)
FILE_NAME:       .ASCII    'BB'              ;LAST 2 CHARS OF FILE NAME

*****
: VARIABLES AREA
*****

SHIFT_CNT:       .BLKB      1                  ;COUNTER FOR WRK REG SHIFTS.
SHIFT_CNT_DAT:   .BYTE      4                  ;COUNTER DATA.
LS_ADDR:         .BYTE      0                  ;LS ADDRESS POINTER.
DATA_PNT:        .WORD      0                  ;TEST E DATA POINTER.
LOOP_CNT:        .BYTE      0                  ;COUNT FOR ^C USE.
SAVE_LS_ADDR:    .BYTE      0                  ;SAVES LS POINTER ON ^C.

MOV_CCR_WRO:     .BYTE      <^X37>             ;INSTRUCTION USED TO PASS DATA
:               .BYTE      <^XF8>             ; FROM THE D-BUS TO WR[0].
:               .BYTE      <^X15>

MOV_WRO_LSO:     .BYTE      <^XBE>             ;INSTRUCTION USED TO PASS DATA
:               .BYTE      <^X00>             ; FORM WR[0] TO LS[0].
:               .BYTE      <^X15>

MOV_LSO_WRO:     .BYTE      <^X36>             ;INSTRUCTION USED TO PASS DATA
:               .BYTE      <^X00>             ; FORM LS[0] TO WR[0].
:               .BYTE      <^X15>

MOV_WRO_LSX:     .BYTE      <^XBE>             ;INSTRUCTION USED TO PASS DATA
:               .BYTE      <^X00>             ; FROM WR[0] TO AN LS LOCATION
:               .BYTE      <^X15>             ; SPECIFIED DURING EXECUTION.

MOV_LSX_WRO:     .BYTE      <^X36>             ;INSTRUCTION USED TO PASS DATA
:               .BYTE      <^X00>             ; FROM AN LS LOCATION SPECIFIED
:               .BYTE      <^X15>             ; DURING EXECUTION TO WR[0].

MOV_WR1_WR1:     .BYTE      <^XA0>             ;INSTRUCTION TO PASS DATA FROM
:               .BYTE      <^X02>             ; WR[1] TO THE Y-BUS.
:               .BYTE      <^X95>

SHIFT_WR1_R:     .BYTE      <^XA3>             ;INSTRUCTION USED TO SHIFT THE
:               .BYTE      <^X40>             ; WR[1] CONTENTS ONE PLACE TO
:               .BYTE      <^X95>             ; THE RIGHT.

```

ZZ-ENKBB-2.0

7000

4

N 1

Fiche 1 Frame N1

Sequence 13

7023 360					
7023 361 23	ROT_WR1_L:	.BYTE	<^X23>		; INSTRUCTION USED TO SHIFT THE
7024 362 C0		.BYTE	<^XC0>		; WR[1] CONTENTS ONE PLACE TO
7025 363 95		.BYTE	<^X95>		; THE LEFT WITH LSB SET.
7026 364					
7026 365 A3	SHIFT_WR1_L:	.BYTE	<^XA3>		; INSTRUCTION USED TO SHIFT THE
7027 366 D0		.BYTE	<^XD0>		; WR[1] CONTENTS ONE PLACE TO
7028 367 95		.BYTE	<^X95>		; THE LEFT WITH LSB CLEAR.
7029 368					
7029 369 2F	X_CLR_WR1:	.BYTE	<^X2F>		; INSTRUCTION USED TO CLEAR
702A 370 82		.BYTE	<^X82>		; WR[1].
702B 371 95		.BYTE	<^X95>		
702C 372					
702C 373 A0	X_COM_WR1:	.BYTE	<^XA0>		; INSTRUCTION USED TO COMPLEMENT
702D 374 C2		.BYTE	<^XC2>		; WR[1].
702E 375 95		.BYTE	<^X95>		
702F 376					
702F 377 10	MISC2_WR_CRR:	.BYTE	<^X10>		; INSTRUCTION USED TO ENABLE
7030 378 FE		.BYTE	<^XFE>		; WR CRR
7031 379 15		.BYTE	<^X15>		
7032 380					
7032 381 2F	SUPPORT_CODE:	.BYTE	<^X2F>		; SIX WCS MICROINSTRUCTIONS USED
7033 382 80		.BYTE	<^X80>		; FOR FAST WRITES OF WCS.
7034 383 15		.BYTE	<^X15>		
7035 384					
7035 385 A0		.BYTE	<^XA0>		; *
7036 386 C0		.BYTE	<^XC0>		
7037 387 15		.BYTE	<^X15>		
7038 388					
7038 389 5B		.BYTE	<^X5B>		; *
7039 390 00		.BYTE	<^X00>		
703A 391 18		.BYTE	<^X18>		
703B 392					
703B 393 A3		.BYTE	<^XA3>		; *
703C 394 D0		.BYTE	<^XD0>		
703D 395 1E		.BYTE	<^X1E>		
703E 396					
703E 397 A3		.BYTE	<^XA3>		; *
703F 398 C0		.BYTE	<^XC0>		
7040 399 15		.BYTE	<^X15>		
7041 400					
7041 401 08		.BYTE	<^X08>		; *
7042 402 00		.BYTE	<^X00>		
7043 403 24		.BYTE	<^X24>		
7044 404					
7044 405 0001	VER_WRITE_TAB:	.WORD	<^X0001>		; *
7046 406 0002		.WORD	<^X0002>		; *
7048 407 0003		.WORD	<^X0003>		; EXPECTED UPC ADDRESSES IN
704A 408 0004		.WORD	<^X0004>		; STEPPING THROUGH THE FAST
704C 409 0006		.WORD	<^X0006>		; WRITE WRO MICROCODE.
704E 410 0003		.WORD	<^X0003>		; *
7050 411 0005		.WORD	<^X0005>		; *
7052 412 0006		.WORD	<^X0006>		; *
7054 413					
7054 414 00	ZERO_DAT:	.BYTE	<^X00>		; *
7055 415 00	ONE_SHIFT:	.BYTE	<^X00>		; *
7056 416 00		.BYTE	<^X00>		; *
7057 417 00		.BYTE	<^X00>		; *
7058 418 01		.BYTE	<^X01>		; STANDARD DATA PATTERNS USED
7059 419 FF	ONE_DAT:	.BYTE	<^XFF>		; BY VARIOUS TESTS.

```

22-ENKBB-2.0      7000      4
705A 420 FF
705B 421 FF
705C 422 FF
705D 423 FE
705E 424
705E 425 FFFF
7060 426 FFFF
7062 427 EEEE
7064 428 EEEE
7066 429 DDDD
7068 430 DDDD
706A 431 BBBB
706C 432 BBBB
706E 433 7777
7070 434 7777
7072 435 0000
7074 436 0000
7076 437 1111
7078 438 1111
707A 439 2222
707C 440 2222
707E 441 4444
7080 442 4444
7082 443 8888
7084 444 8888
7086 445
7086 446
7086 447
7086 448
7086 449
7086 450
7086 451
7086 452
7086 453
7086 454 03
7087 455 0000008A
708A 456
708A 457 04
708B 458 0000008F
708F 459
708F 460 04
7090 461 00000094
7094 462
7094 463 04
7095 464 00000099
7099 465
7099 466
7099 467
7099 468
7099 469
7099 470
7099 471
7099 472
7099 473
7099 474
7099 475
7099 476 0009'3A
709C 477 000D'32
709F 478
709F 479 0000'CD

```

```

      B 2
ZERO_SHIFT: .BYTE <^XFF>
              .BYTE <^XFF>
              .BYTE <^XFF>
              .BYTE <^XFE>

TESTE_DAT:  .WORD <^XFFFF>
              .WORD <^XFFFF>
              .WORD <^XEEEE>
              .WORD <^XEEEE>
              .WORD <^XDDDD>
              .WORD <^XDDDD>
              .WORD <^XDDDD>
              .WORD <^X8888>
              .WORD <^X8888>
              .WORD <^X7777>
              .WORD <^X7777>
              .WORD <^X0000>
              .WORD <^X0000>
              .WORD <^X0000>
              .WORD <^X1111>
              .WORD <^X1111>
              .WORD <^X2222>
              .WORD <^X2222>
              .WORD <^X4444>
              .WORD <^X4444>
              .WORD <^X88C8>
              .WORD <^X8888>

```

Fiche 1 Frame B2

Sequence 14

DATA TO BE WRITTEN TO LS
CONSISTS OF ALL ZEROS,
ALL ONES, AND 4 BIT BLOCKS
OF A ONE THROUGH ZEROS
AND A ZERO THROUGH ONES.

```

*****
:
:  SHIFT VARIABLES - DO NOT SEPARATE PAIRS
:
*****

```

```

TEMPA_SHIFT: .BYTE 3 ;FIRST TEMPORARY STORAGE LOCATION
TEMPA_DAT:   .BLKB 3 ; FOR DATA FROM VARIOUS TESTS.

TEMPB_SHIFT: .BYTE 4 ;SECOND TEMPORARY STORAGE LOCATION
TEMPB_DAT:   .BLKB 4 ; FOR DATA FROM VARIOUS TESTS.

EXP_SHIFT:   .BYTE 4 ;LOCATION OF EXPECTED DATA FROM
EXP_DAT:     .BLKB 4 ; VARIOUS TESTS.

REC_SHIFT:   .BYTE 4 ;LOCATION OF RETURNED DATA FROM
REC_DAT:     .BLKB 4 ; VARIOUS TESTS.

```

```

*****
:
:  CON_SET_FOR_CO THIS ROUTINE IS USED IF A CONTROL C OCCURS TO SAVE THE
:                  STATE OF THE MACHINE. IF A CONTINUE IS THEN ISSUED, THIS
:                  ROUTINE RESTORES THE STATE OF THE MACHINE.
:
*****

```

```

CON_SET_FOR_CO: LDA    LS_ADDR    ;SAVE THE PRESENT LS ADDRESS.
                  STA    SAVE_LS_ADDR
                  CALL   SET_FOR_CONT ;RETURN TO MONITOR.

```

ZZ-ENKBB-2.0 7000 4

```

70A2 480
70A2 481 000D'3A
70A5 482 0009'32
70A8 483
70A8 484 C9
70A9 485
70A9 486
70A9 487
70A9 488
70A9 489
70A9 490
70A9 491
70A9 492
70A9 493 40D9 3A
70AC 494 B7
70AD 495 C8
70AE 496 0000'CD
70B1 497
70B1 498 0000'3A
70B4 499 0F
70B5 500 0099'DC
70B8 501
70B8 502 C9
70B9 503
70B9 504
70B9 505
70B9 506
70B9 507
70B9 508
70B9 509
70B9 510
70B9 511
70B9 512
70B9 513
70B9 514 00'32'21
70BC 515 000A'22
70BF 516
70BF 517 00 00 21
70C2 518 0000'22
70C5 519
70C5 520 C5 46 00'00'21
              77 06 3E
70CD 521
70CD 522
70CD 523 00A9'CD
70D0 524
70D0 525 000A'2A
70D3 526 00'00'11
70D6 527 0000'CD
70D9 528
70D9 529 0000'CD
70DC 530
70DC 531 00'00'21
70DF 532 0000'CD
70E2 533
70E2 534 000A'2A
70E5 535 23
70E6 536 23
70E7 537 23
70E8 538 000A'22

```

```

LDA  SAVE_LS_ADDR  ;RESTORE STATE.
STA  LS_ADDR
RET

```

```

*****
: CHECK_RD  THIS ROUTINE CHECKS TO SEE IF DIAGNOSTIC IS UNDER RD CONTROL.
:           IF IT IS, A CALL IS MADE TO GCVECT SO THAT THE BOOT CODE CAN
:           KEEP TRACK OF ELAPSED TIME.
*****

```

```

CHECK_RD:  LDA  MIPFLG      ;GET STATE OF MODEM IN PROGRESS
           ORA  A          ;SET CONDITION CODES
           RZ           ;RETURN IF ZERO
           CALL GCVECT     ;ELSE GO TO BOOT CODE IF RD

           LDA  CNTLC_TYPED ;SEE IF CONTROL C TYPED (UNDER RD)
           RRC
           CC  CON_SET_FOR_CO ;IF SO, GO TO MONITOR.

           RET             ;BACK TO TEST

```

```

*****
: LD_SUPPORT THIS ROUTINE LOADS SUPPORT CODE INTO WCS LOCATIONS 0-13 TO
:           ENABLE FAST READS AND WRITES OF WORKING REGISTER 0 AND THUS
:           OF LOCAL STORE.
*****

```

```

LD_SUPPORT: LXI  H,SUPPORT_CODE ;POINT TO FIRST SUPPORT MICROWORD.
           SHLD DATA_PNT

           LXI  H,0             ;POINT TO WCS ADDRESS ZERO.
           SHLD WCS_ADDRESS

           DO 6                  ;LOAD FIRST SIX SUPPORT MICROWORDS
           ; FOR WRITING DATA TO WRO.

           CALL CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL

           LHLD DATA_PNT       ;MOVE A MICROWORD TO WCS_VALUE.
           LXI  D,WCS_VALUE
           CALL MOVER_3

           CALL WCS_WRITE       ;WRITE IT TO WCS.

           LXI  H,WCS_ADDRESS   ;POINT TO THE NEXT WCS LOCATION.
           CALL INC_WORD

           LHLD DATA_PNT       ;POINT TO THE NEXT SUPPORT MICROWORD.
           INX  H
           INX  H
           INX  H
           SHLD DATA_PNT

```


ZZ-ENKBB-2.0 7000 4

D 2

Fiche 1 Frame D2

Sequence 16

```

70EB 539 *
70EB 540 35 00'00'21 *****
      70 C1 00CD'C2
70F4 541 *
70F4 542 C5 46 00'00'21 *****
      77 08 3E
70FC 543 *
70FC 544 *
70FC 545 00A9'CD *
70FF 546 *
70FF 547 00'01'21 *
7102 548 00'00'11 *
7105 549 0000'CD *
7108 550 *
7108 551 0000'CD *
7108 552 *
7108 553 00'00'21 *
710E 554 0000'CD *
7111 555 *
7111 556 35 00'00'21 *****
      70 C1 00FC'C2
711A 557
711A 558 C9
711B 559
711B 560
711B 561
711B 562
711B 563
711B 564
711B 565
711B 566
711B 567
711B 568
711B 569
711B 570
711B 571 0000'CD
711E 572 0000'2A
7121 573 0097'22
7124 574 0000'CD
7127 575
7127 576 0000'2A
712A 577 7E
712B 578 0093'32
712E 579 23
712F 580 7E
7130 581 0092'32
7133 582
7133 583 23
7134 584 0000'22
7137 585
7137 586 C9
7138 587
7138 588
7138 589
7138 590
7138 591
7138 592
7138 593
7138 594
7138 595

```

ENDDO

```

DO      8      ;LOAD THE NEXT 8 MICROWORDS FOR
           ; READING WR[0].
CALL    CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL
LXI     H,X_SHIFT_R+1 ;LOAD A SHIFT RIGHT MICROWORD TO
LXI     D,WCS_VALUE  ; WCS_VALUE.
CALL    MOVER_3
CALL    WCS_WRITE    ;WRITE IT TO WCS.
LXI     H,WCS_ADDRESS ;POINT TO THE NEXT WCS LOCATION.
CALL    INC_WORD
ENDDO

```

RET

```

*****
: SET_COMPARE THIS ROUTINE READS THE UPC AND PUTS IT IN REC_DAT+2 AND READS
: THE CORRESPONDING VERIFY WRITER TABLE ENTRY AND PUTS IT IN
: EXP_DAT+2. THE TABLE POINTER IS INCREMENTED.
*****

```

```

SET_COMPARE: CALL    LOAD_UPC      ;PUT THE UPC CONTENTS IN REC_DAT.
              LHLD    UPC_VALUE
              SHLD    REC_DAT+2
              CALL    LOAD_UPC      ;AND RELOAD THE UPC.
              LHLD    WRITE_TAB_PNT ;LOAD AN ENTRY FROM THE VERIFY WRITE
              MOV     A,M             ; WRITE TABLE INTO EXP_DAT.
              STA     EXP_DAT+3
              INX     H
              MOV     A,M
              STA     EXP_DAT+2
              INX     H               ;INCREMENT THE VERIFY WRITE TABLE
              SHLD    WRITE_TAB_PNT ; POINTER.
              RET

```

```

*****
: LS_WRITE THIS ROUTINE WRITES FOUR BYTES OF DATA FROM DATA0-DATA3 TO LOCAL
: STORE AT THE LOCATION GIVEN BY THE CONTENTS OF THE ACCUMULATOR.
*****

```

ZZ-ENKBB-2.0 7000 4

E 2

Fiche 1 Frame E2

Sequence 17

7138 596
7138 597 0000'32
713B 598
713B 599 80 E6
713D 600 07
713E 601 BE F6
7140 602 0017'32
7143 603
7143 604
7143 605 0000'3A
7146 606 B7
7147 607 17
7148 608 0018'32
714B 609
714B 610
714B 611 0172'CD
714E 612
714E 613 00'17'21
7151 614 0294'CD
7154 615
7154 616 C9
7155 617
7155 618
7155 619
7155 620
7155 621
7155 622
7155 623
7155 624
7155 625
7155 626
7155 627 0000'32
7158 628
7158 629 80 E6
715A 630 07
715B 631 36 F6
715D 632 001A'32
7160 633
7160 634
7160 635 0000'3A
7163 636 B7
7164 637 17
7165 638 001B'32
7168 639
7168 640
7168 641 00'1A'21
716B 642 0294'CD
716E 643
716E 644 01B9'CD
7171 645
7171 646 C9
7172 647
7172 648
7172 649
7172 650
7172 651
7172 652
7172 653
7172 654
7172 655

```
LS_WRITE:  STA      XD_ADDRS      ;STORE THE LS ADDRESS IN XD_ADDRS.
           ANI      <^X80>        ;SAVE THE MSB.
           RLC      ;ROTATE LEFT ONE PLACE.
           ORI      <^XBE>        ;OR IN THE MOV WRO TO LS OP CODE.
           STA      MOV_WRO_LSX   ;STORE BIT 7 OF EXTENDED ADDRESS IN
                                   ; BIT 16 OF INSTRUCTION.

           LDA      XD_ADDRS      ;LOAD EXTENDED ADDRESS.
           ORA      A              ;CLEAR THE CARRY.
           RAL      ;SHIFT LEFT TO CLEAR LSB (B FIELD).
           STA      MOV_WRO_LSX+1 ;STORE BITS 0-6 OF EXTENDED ADDRESS
                                   ; IN BITS 9-15 OF INSTRUCTION.

           CALL     FAST_WRITE_WRO ;WRITE DATA TO WORKING REG 0.

           LXI      H,MOV_WRO_LSX ;MOVE THE DATA FROM WRO TO LS.
           CALL     EXEC_INST

           RET
```

```
*****
: LS_READ THIS ROUTINE READS FOUR BYTES OF DATA FROM THE LS LOCATION
: POINTED TO BY THE ACCUMULATOR TO DATA0-DATA3.
*****
```

```
LS_READ:  STA      XD_ADDRS      ;STORE THE LS ADDRESS IN XD_ADDRS.
           ANI      <^X80>        ;SAVE THE MSB.
           RLC      ;ROTATE LEFT ONE PLACE.
           ORI      <^X36>        ;OR IN THE MOV WRO TO LS OP CODE.
           STA      MOV_LSX_WRO   ;STORE BIT 7 OF EXTENDED ADDRESS IN
                                   ; BIT 16 OF INSTRUCTION.

           LDA      XD_ADDRS      ;LOAD EXTENDED ADDRESS.
           ORA      A              ;CLEAR THE CARRY.
           RAL      ;SHIFT LEFT TO CLEAR LSB (B FIELD).
           STA      MOV_LSX_WRO+1 ;STORE BITS 0-6 OF EXTENDED ADDRESS
                                   ; IN BITS 9-15 OF INSTRUCTION.

           LXI      H,MOV_LSX_WRO ;EXECUTE THE INSTRUCTION.
           CALL     EXEC_INST

           CALL     FAST_READ_WRO  ;READ WRO.

           RET
```

```
*****
: FAST_WRITE_WRO THIS ROUTINE PERFORMS A FAST WRITE OF WORKING REGISTER
: ZERO BY USING SUPPORT MICROCODE IN WCS.
*****
```

```

FAST_WRITE_WRO: LXI      H,0          ;PREPARE TO EXECUTE WCS SUPPORT CODE
                SHLD     UPC_VALUE    ; STORED IN WCS LOCATIONS 0-5.
                CALL     LOAD_UPC

                CALL     NOP_CSR      ;LOAD CSR WITH NOP INSTRUCTION.

                OUT      SETSS        ;EXECUTE WCS INSTRUCTION TO CLEAR WRO.
                OUT      CLRSS

                OUT      SETSS        ;COMPLEMENT WRO.
                OUT      CLRSS

                CALL     CHECK_RD      ;CALL BOOT CODE IF UNDER RD CONTROL
                DO       32

                LXI      H,DATA_SHIFT+4 ;POINT TO END OF DATA AREA.
                MVI      C,4          ;SHIFT FOUR BYTES INTO WRO.
                CALL     SHIFTER1

                OUT      CLRATN       ;SET A ZERO TO BE READ.

                IFC
                OUT      SETATN       ;SET A ONE TO BE READ.
                ENDIF

                OUT      SETSS        ;EXECUTE WCS SUBROUTINE TO SHIFT
                OUT      CLRSS        ; A BIT INTO WRO.

                OUT      SETSS        ;*
                OUT      CLRSS

                OUT      SETSS        ;*
                OUT      CLRSS

                ENDDO

                CALL     CHECK_RD      ;CALL BOOT CODE IF UNDER RD CONTROL
                RET

;*****
; FAST_READ_WRO THIS ROUTINE PERFORMS A FAST READ OF WORKING REGISTER ZERO
; USING SUPPORT MICROCODE IN WCS.
;*****

FAST_READ_WRO: LXI      H,X_MOV_WRWR ;ENABLE WRO TO BE READ.
                CALL     LD_CSR_INST

                OUT      YBUSRD       ;LATCH THE DATA IN THE Y-BUS LATCH.
                IN       READ         ;READ THE Y-BUS.
                STA      DATA3       ;STORE IN DATA3.

```

```

ZZ-ENKBB-2.0 7000 4
71C6 714 01D9'CD
71C9 715 0000'32
71CC 716
71CC 717 01D9'CD
71CF 718 0000'32
71D2 719
71D2 720 01D9'CD
71D5 721 0000'32
71D8 722
71D8 723 C9
71D9 724
71D9 725
71D9 726
71D9 727
71D9 728
71D9 729
71D9 730
71D9 731
71D9 732
71D9 733
71D9 734
71D9 735
71D9 736 06 00 21
71DC 737 0000'22
71DF 738 0000'CD
71E2 739
71E2 740 0000'CD
71E5 741
71E5 742 C5 46 00'00'21
71ED 743 77 09 3E
71ED 744 23 D3
71EF 745 22 D3
71F1 746
71F1 747 35 00'00'21
71FA 748 70 C1 01ED'C2
71FA 749 00'00'21
71FD 750 029C'CD
7200 751
7200 752 EC D3
7202 753 80 DB
7204 754
7204 755 C9
7205 756
7205 757
7205 758
7205 759
7205 760
7205 761
7205 762
7205 763
7205 764
7205 765
7205 766 00'29'21
7208 767 0294'CD
7208 768
7208 769 00'2C'21
720E 770 0294'CD
7211 771

```

G 2

Fiche 1 Frame G2 Sequence 19

```

CALL READ_BYTE_WRO ;READ THE NEXT BYTE OF WRO.
STA DATA2 ;STORE IN DATA2.

```

```

CALL READ_BYTE_WRO ;AND THE NEXT...
STA DATA1

```

```

CALL READ_BYTE_WRO ;AND THE LAST.
STA DATA0

```

RET

```

*****
: READ_BYTE_WRO THIS ROUTINE EXECUTES SUPPORT CODE OUT OF WCS WHICH SHIFTS
: THE CONTENTS OF WORKING REGISTER ZERO TO THE RIGHT EIGHT
: PLACES. IT THEN READS THE LOW BYTE OF WRO AND LEAVES IT
: IN THE ACCUMULATOR.
*****

```

```

READ_BYTE_WRO: LXI H,<^X0600> ;PREPARE TO EXECUTE WCS SUPPORT CODE
SHLD UPC_VALUE ; STORED IN WCS LOCATIONS 6-D.
CALL LOAD_UPC

```

```

CALL NOP_CSR ;LOAD CSR WITH NOP INSTRUCTION.

```

```

DO 9

```

```

OUT SETSS ;EXECUTE A SHIFT INSTRUCTION.
OUT CLRSS

```

ENDDO

```

LXI H,X MOV WRWR ;ENABLE WRO TO BE READ.
CALL LD_CSR_INST

```

```

OUT YBUSRD ;LATCH THE DATA IN THE Y-BUS LATCH.
IN READ ;READ THE Y-BUS.

```

RET

```

*****
: WRITE_WR1_32 THIS ROUTINE LOADS 4 BYTES OF DATA FROM DATA0 - DATA3 TO THE
: 2901'S WR[1].
*****

```

```

WRITE_WR1_32: LXI H,X CLR WR1 ;CLEAR WR1.
CALL EXEC_INST

```

```

LXI H,X COM WR1 ;COMPLEMENT WR1.
CALL EXEC_INST

```

22-ENKBB-2.0 7000 4

```

7211 772 C5 46 00'00'21
      77 20 3E
7219 773
7219 774 00'00'21
721C 775 0000'22
721F 776
721F 777 0000'CD
7222 778
7222 779 022E'D2
7225 780
7225 781 00'23'21
7228 782 0294'CD
722B 783
722B 784 0234'C3
722E 785
722E 786 00'26'21
7231 787 0294'CD
7234 788
7234 789
7234 790
7234 791 35 00'00'21
      70 C1 0219'C2
723D 792
723D 793 C9
723E 794
723E 795
723E 796
723E 797
723E 798
723E 799
723E 800
723E 801
723E 802
723E 803
723E 804 00'1D'21
7241 805 029C'CD
7244 806
7244 807 EC D3
7246 808 80 DB
7248 809 0000'32
724B 810 027C'CD
724E 811
724E 812 00'1D'21
7251 813 029C'CD
7254 814
7254 815 EC D3
7256 816 80 DB
7258 817 0000'32
725B 818 027C'CD
725E 819
725E 820 00'1D'21
7261 821 029C'CD
7264 822
7264 823 EC D3
7266 824 80 DB
7268 825 0000'32
726B 826 027C'CD
726E 827
726E 828 00'1D'21
7271 829 029C'CD

```

H 2

Fiche 1 Frame H2

Sequence 20

```

DO 32
LXI H,DATA_SHIFT ;PREPARE TO SHIFT DATA INTO WR1.
SHLD SHIFT_PNT
CALL SHIFTER ;SHIFT A BIT OF DATA INTO THE CARRY.
IFC ;IF THE CARRY IS SET...
LXI H,ROT_WR1_L ;...SHIFT A ONE INTO WR1.
CALL EXEC_INST
ELSE ;OTHERWISE...
LXI H,SHIFT_WR1_L ;...SHIFT A ZERO INTO WR1.
CALL EXEC_INST
ENDIF
ENDDO
RET

```

```

*****
: READ_WR1_32 THIS ROUTINE IS USED TO READ THE CONTENTS OF THE 2901'S
: WR[1]. THE DATA IS PUT INTO LOCATIONS DATA0 - DATA3.
*****

```

```

READ_WR1_32: LXI H,MOV_WR1_WR1 ;LOAD THE CSR WITH AN INSTRUCTION
              CALL LD_CSR_INST ; TO ENABLE WR1 TO BE READ.
              OUT YBUSRD ;READ THE LOW BYTE AND SHIFT
              IN READ ; THE CONTENTS OF WR1 TO THE
              STA DATA3 ; RIGHT EIGHT BITS.
              CALL SHIFT_WR1_R8
              LXI H,MOV_WR1_WR1 ;REPEAT FOR NEXT BYTE.
              CALL LD_CSR_INST
              OUT YBUSRD
              IN READ
              STA DATA2
              CALL SHIFT_WR1_R8
              LXI H,MOV_WR1_WR1 ;REPEAT FOR NEXT BYTE.
              CALL LD_CSR_INST
              OUT YBUSRD
              IN READ
              STA DATA1
              CALL SHIFT_WR1_R8
              LXI H,MOV_WR1_WR1 ;REPEAT FOR LAST BYTE.
              CALL LD_CSR_INST

```

22-ENKBB-2.0 7000 4

I 2

Fiche 1 Frame I2

Sequence 21

7274 830
7274 831 EC D3
7276 832 80 DB
7278 833 0000'32
7278 834
7278 835 C9
727C 836
727C 837
727C 838
727C 839
727C 840
727C 841
727C 842
727C 843
727C 844
727C 845
727C 846 C5 46 00'00'21
77 08 3E
7284 847
7284 848 00'20'21
7287 849 0294'CD
728A 850
728A 851 35 00'00'21
70 C1 0284'C2
7293 852
7293 853 C9
7294 854
7294 855
7294 856
7294 857
7294 858
7294 859
7294 860
7294 861
7294 862
7294 863
7294 864 029C'CD
7297 865
7297 866 23 D3
7299 867 22 D3
7298 868
7298 869 C9
729C 870
729C 871
729C 872
729C 873
729C 874
729C 875
729C 876
729C 877
729C 878
729C 879
729C 880 00'00'11
729F 881 0000'CD
72A2 882
72A2 883 A2 D3
72A4 884
72A4 885 00'00'21
72A7 886 0000'22
72AA 887

*
*
*
*

SHIFT_WR1_R8: DO 8

LXI H,SHIFT_WR1_R ;SHIFT WR1 RIGHT ONE BIT.
CALL EXEC_INST

ENDDO

RET

EXEC_INST THIS ROUTINE EXECUTES AN INSTRUCTION POINTED TO BY THE H,L
PAIR.

EXEC_INST: CALL LD_CSR_INST ;LOAD INSTRUCTION INTO THE CSR.

OUT SETSS ;CLOCK THE CPU.
OUT CLRSS

RET

LD_CSR_INST THIS ROUTINE LOADS THE CSR WITH AN INSTRUCTION, WHICH IS
POINTED TO BY THE CONTENTS OF THE H,L PAIR.

LD_CSR_INST: LXI D,CSR_VALUE ;MOVE INSTRUCTION POINTED TO BY THE
CALL MOVER_3 ; H,L PAIR INTO CSR_VALUE.

OUT VSHIFT ;SET V-BUS TO SHIFTING MODE.

LXI H,CSR_SHIFT ;PREPARE TO SHIFT DATA INTO CSR.
SHLD SHIFT_PNT

22-ENKBB-2.0	7000	4
72AA 888	C5 46 00'00'21	
	77 18 3E	
72B2 889		
72B2 890	38 D3	
72B4 891		
72B4 892	87 DB	
72B6 893	1F	
72B7 894	0000'CD	
72BA 895		
72BA 896	02BF'D2	
72BD 897		
72BD 898	39 D3	
72BF 899		
72BF 900	25 D3	
72C1 901	24 D3	
72C3 902		
72C3 903	35 00'00'21	
	70 C1 02B2'C2	
72CC 904		
72CC 905	A3 D3	
72CE 906		
72CE 907	00A9'CD	
72D1 908		
72D1 909	C9	
72D2 910		
72D2 911		
72D2 912		
72D2 913		
72D2 914		
72D2 915		
72D2 916		
72D2 917		
72D2 918		
72D2 919		
72D2 920	00'00'11	
72D5 921	05 0E	
72D7 922	0000'CD	
72DA 923		
72DA 924	C9	
72DB 925		
72DB 926		
72DB 927		
72DB 928		
72DB 929		
72DB 930		
72DB 931		
72DB 932		
72DB 933		
72DB 934	0090'32	
72DE 935	CC D3	
72E0 936		
72E0 937	23 D3	
72E2 938	22 D3	
72E4 939		
72E4 940	00'00'21	
72E7 941	029C'CD	
72EA 942		
72EA 943	EC D3	
72EC 944	80 DB	
72EE 945	0095'32	

15:

J 2

DO 24

Fiche 1 Frame J2

Sequence 22

```

OUT      CLRCSR          ;CLEAR THE CSR SERIAL INPUT.

IN       CSR23           ;READ THE CSR'S MSB.
RAR      ;ROTATE IT INTO THE CARRY.
CALL     SHIFTER         ;SHIFT THE DATA THROUGH CSR_VALUE.

JNC      1$              ;IF THE CARRY IS SET...

OUT      SETCSR          ;SET THE CSR SERIAL INPUT.

OUT      SETCSK          ;CLOCK THE CSR SERIAL LOAD.
OUT      CLRCSK

ENDDO

OUT      VNORML          ;SET V-BUS TO NORMAL MODE.

CALL     CHECK_RD        ;CALL BOOT CODE IF RD IN CONTROL

RET

```

```

;*****
;
;  LOAD_MOD_DAT  THIS ROUTINE LOADS THE MODULE DATA POINTED TO BY THE H,L
;                PAIR INTO CON_MOD_DAT.
;
;*****

```

```

LOAD_MOD_DAT:  LXI      D,CON_MOD_DAT      ;MOVE THE MODULE DATA POINTED TO BY
               MVI      C,5                ; THE H,L PAIR TO CON_MOD_DAT.
               CALL     MOVER
               RET

```

```

:*****
:
:  TEST9_1_0    THIS ROUTINE IS USED BY TEST 9 TO WRITE AN 8-BIT DATA PATTERN
:                INTO THE ALU AND THEN READ IT BACK
:*****

```

TEST9_1_0:	STA OUT	EXP_DAT WRITE	;LOAD DATA PATTERN INTO EXP_DAT. ;WRITE THIS PATTERN TO THE D-BUS.
	OUT OUT	SETSS CLRSS	;CLOCK THE CPU TO ENABLE WR OUTPUT.
	LXI CALL	H,X MOV WRWR LD_CSR_INST	;LOAD THE CSR WITH AN INSTRUCTION TO ; ENABLE THE WR TO BE READ.
	OUT IN STA	YBUSRD READ REC_DAT	;CLOCK THE Y-BUS LATCH. ;READ THE Y-BUS CONTENTS. ;STORE THE Y-BUS CONTENTS IN REC_DAT.

ZZ-ENKBB-2.0 7000 4

72F1 946
72F1 947 00'90'21
01 0E 00'95'11
0302'CA 0000'CD

72FF 948
72FF 949
72FF 950 037B'CD

7302 951
7302 952
7302 953
7302 954 0000'3A
7305 955 0F
7306 956 C9

7307 957
7307 958
7307 959

7307 960
7307 961
7307 962

7307 963
7307 964
7307 965

7307 966
7307 967 00'0E'21
730A 968 029C'CD

730D 969
730D 970 0087'3A
7310 971 0090'32

7313 972 CC D3
7315 973
7315 974 23 D3

7317 975 22 D3
7319 976
7319 977 00'00'21

731C 978 029C'CD
731F 979
731F 980 EC D3

7321 981 80 DB
7323 982 0095'32
7326 983

7326 984 00'90'21
01 0E 00'95'11
0337'CA 0000'CD

7334 985
7334 986
7334 987 037B'CD

7337 988
7337 989
7337 990

7337 991 0000'3A
733A 992 0F
733B 993 0307'DA
733E 994 C9

733F 995
733F 996
733F 997

733F 998
733F 999
733F 1000

733F 1001

K 2

Fiche 1 Frame K2

Sequence 23

IFNEQ EXP_DAT,REC_DAT,1 ;COMPARE THE EXPECTED AND RECEIVED

; DATA.

CALL TEST9_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC ;RETURN WITH C SET IF LOOP ON ERROR
RET ; AND C CLEAR IF NO LOOP

: TEST9_SHIFT THIS ROUTINE IS USED BY TEST 9 TO WRITE AN 8-BIT SHIFT PATTERN
: INTO THE ALU AND THEN READ IT BACK
: *****

TEST9_SHIFT: LXI H,MOV_CCR_WRO ;LOAD THE CSR WITH A MOVE INSTRUCTION
CALL LD_CSR_INST ; USED TO PASS DATA THROUGH THE ALU.

LDA TEMPA_DAT ;LOAD THE DATA INTO A AND INTO
STA EXP_DAT ; EXP DAT.
OUT WRITE ;WRITE THIS PATTERN TO THE D-BUS.

OUT SETSS ;CLOCK THE CPU TO ENABLE WR OUTPUT.
OUT CLRSS

LXI H,X_MOV_WRWR ;LOAD THE CSR WITH AN INSTRUCTION TO
CALL LD_CSR_INST ; ENABLE THE WR TO BE READ.

OUT YBUSRD ;CLOCK THE Y-BUS LATCH.
IN READ ;READ THE Y-BUS CONTENTS.
STA REC_DAT ;STORE THE Y-BUS CONTENTS IN REC_DAT.

IFNEQ EXP_DAT,REC_DAT,1 ;COMPARE THE EXPECTED AND RECEIVED

; DATA.

CALL TEST9_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC TEST9_SHIFT ;...JUMP TO BEGINNING OF ERROR LOOP.
RET

: TEST9_2_0 THIS ROUTINE IS USED BY TEST 9 TO WRITE AN 8 BIT DATA PATTERN
: INTO THE ALU AND THEN READ IT BACK WHILE TESTING WR CRR
: *****

ZZ-ENKBB-2.0 7000 4

L 2

Fiche 1 Frame L2

Sequence 24

733F 1002
733F 1003
733F 1004
733F 1005 0090'32
7342 1006
7342 1007 00'0E'21
7345 1008 029C'CD
7348 1009
7348 1010 0090'3A
734B 1011
734B 1012 CC D3
734D 1013
734D 1014 23 D3
734F 1015 22 D3
7351 1016
7351 1017 00'2F'21
7354 1018 029C'CD
7357 1019
7357 1020 23 D3
7359 1021 22 D3
735B 1022
735B 1023 80 DB
735D 1024
735D 1025 0095'32
7360 1026
7360 1027 00'90'21
01 0E 00'95'11
0376'CA 0000'CD
736E 1028
736E 1029
736E 1030 02 3E
7370 1031 0000'32
7373 1032
7373 1033 0380'CD
7376 1034
7376 1035
7376 1036
7376 1037 0000'3A
7379 1038 0F
737A 1039 C9
737B 1040
737B 1041
737B 1042
737B 1043
737B 1044
737B 1045
737B 1046
737B 1047
737B 1048
737B 1049
737B 1050
737B 1051 01 3E
737D 1052 0000'32
7380 1053
7380 1054 0090'3A
7383 1055 0003'32
7386 1056
7386 1057 0095'3A
7389 1058 0003'32
738C 1059

*
*
*
*
*
*

```

;*****
TEST9_2_0:  STA    EXP_DAT      ;LOAD PATTERN INTO EXP_DAT
            LXI    H,MOV_CCR_WRO ;LOAD THE CSR WITH A MOVE INSTRUCTION
            CALL   LD_CSR_INST  ; USED TO PASS DATA THROUGH THE ALU
            LDA    EXP_DAT
            OUT    WRITE        ;WRITE PATTERN TO THE D-BUS
            OUT    SETSS        ;CLOCK DATA PATH
            OUT    CLRSS
            LXI    H,MISC2_WR_CRR ;LOAD THE CSR WITH A MISC INSTRUCTION
            CALL   LD_CSR_INST  ; USED TO ENABLE WR_CRR
            OUT    SETSS        ;CLOCK DATA PATH
            OUT    CLRSS
            IN     READ         ;READ THE Y-BUS CONTENTS
            STA    REC_DAT      ;STORE THE Y-BUS CONTENTS IN REC_DAT
            IFNEQ  EXP_DAT,REC_DAT,1 ;COMPARE THE EXPECTED AND RECEIVED
                                   ; DATA
            MVI    A,2          ;ERROR NUMBER 2
            STA    CON_ERR_NUM
            CALL   TEST9_ERR2_3 ;REPORT ERROR
            ENDF
            LDA    ERROR_CON     ;IF THE ERROR LOOPING FLAG IS SET...
            RRC                ;RETURN WITH C SET IF LOOP ON ERROR
            RET                 ; AND C CLEAR IF NO LOOP

```

```

;*****
; TEST9_ERROR THIS ROUTINE IS USED BY TEST 9 TO DEFINE THE ERROR VARIABLES
; IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
;*****

```

```

TEST9_ERROR:  MVI    A,1          ;ERROR NUMBER 1.
              STA    CON_ERR_NUM

TEST9_ERR2_3: LDA    EXP_DAT      ;MOVE EXPECTED DATA TO CON_EXP_DAT.
              STA    CON_EXP_DAT+3
              LDA    REC_DAT      ;MOVE RECEIVED DATA TO CON_REC_DAT.
              STA    CON_REC_DAT+3

```

```

22-ENKBB-2.0 7000 4
738C 1060 FF FF 21
738F 1061 0000'22
7392 1062 00 FF 21
7395 1063 0002'22
7398 1064
7398 1065 0000'32 3C AF
739D 1066
739D 1067 0446'C3
73A0 1068
73A0 1069
73A0 1070
73A0 1071
73A0 1072
73A0 1073
73A0 1074
73A0 1075
73A0 1076
73A0 1077
73A0 1078 0090'3A
73A3 1079 0003'32
73A6 1080
73A6 1081 0095'3A
73A9 1082 0003'32
73AC 1083
73AC 1084 FF FF 21
73AF 1085 0000'22
73B2 1086 00 FF 21
73B5 1087 0002'22
73B8 1088
73B8 1089 03 0E 00'00'21
          0D 23 77 AF
          03BE'C2
73C4 1090 0008'3A
73C7 1091 0003'32
73CA 1092
73CA 1093 0446'C3
73CD 1094
73CD 1095
73CD 1096
73CD 1097
73CD 1098
73CD 1099
73CD 1100
73CD 1101
73CD 1102
73CD 1103
73CD 1104 00'90'21
73D0 1105 00'00'11
73D3 1106 0000'CD
73D6 1107
73D6 1108 00'95'21
73D9 1109 00'00'11
73DC 1110 0000'CD
73DF 1111
73DF 1112 0446'C3
73E2 1113
73E2 1114
73E2 1115
73E2 1116
73E2 1117

```

M 2

Fiche 1 Frame M2 Sequence 25

```

LXI H,<^XFFFF> ;ONLY THE LAST BYTE IS OF
SHLD CON_MSK_DAT ; INTEREST.
LXI H,<^X00FF>
SHLD CON_MSK_DAT+2

SET NA_OTHER ;NO ADDITIONAL DATA.

JMP REPORT_CPU_ERROR ;REPORT ERROR ON CPU BOARD

```

```

:*****
:
: TESTA_ERROR THIS ROUTINE IS USED BY TEST A TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
:*****

```

```

TESTA_ERROR: LDA EXP_DAT ;MOVE EXPECTED DATA TO CON_EXP_DAT.
              STA CON_EXP_DAT+3
              LDA REC_DAT ;MOVE RECEIVED DATA TO CON_REC_DAT.
              STA CON_REC_DAT+3
              LXI H,<^XFFFF> ;ERROR MASK INCLUDES ONLY THE LEAST
              SHLD CON_MSK_DAT ; SIGNIFICANT BYTE.
              LXI H,<^X00FF>
              SHLD CON_MSK_DAT+2
              ZERO CON_ADD_DAT,3 ;ADDITIONAL DATA CONTAINS THE SHIFT

              LDA SHIFT_CNT_DAT ; DEPTH.
              STA CON_ADD_DAT+3

              JMP REPORT_CPU_ERROR ;REPORT ERROR ON CPU BOARD

```

```

:*****
:
: TESTB_ERROR THIS ROUTINE IS USED BY TEST B TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
:*****

```

```

TESTB_ERROR: LXI H,EXP_DAT ;MOVE EXPECTED DATA TO CON_EXP_DAT.
              LXI D,CON_EXP_DAT
              CALL MOVER_4
              LXI H,REC_DAT ;MOVE RECEIVED DATA TO CON_REC_DAT.
              LXI D,CON_REC_DAT
              CALL MOVER_4

              JMP REPORT_CPU_ERROR ;REPORT ERROR ON CPU BOARD

```

```

:*****
:
: TESTD_ERROR THIS ROUTINE IS USED BY TEST D TO DEFINE THE ERROR VARIABLES

```

```

ZZ-ENKBB-2.0    7000    4
73E2 1118
73E2 1119
73E2 1120
73E2 1121
73E2 1122
73E2 1123
73E2 1124
73E2 1125 01 3E
73E4 1126 0000'32
73E7 1127
73E7 1128 00'90'21
73EA 1129 00'00'11
73ED 1130 0000'CD
73F0 1131
73F0 1132 00'95'21
73F3 1133 00'00'11
73F6 1134 0000'CD
73F9 1135
73F9 1136 0000'32 3C AF
73FE 1137
73FE 1138 04 0E 00'00'21
              0D 23 77 AF
              0404'C2
740A 1139
740A 1140 0446'C3
740D 1141
740D 1142
740D 1143
740D 1144
740D 1145
740D 1146
740D 1147
740D 1148
740D 1149
740D 1150
740D 1151 00'90'21
7410 1152 00'00'11
7413 1153 0000'CD
7416 1154
7416 1155 00'00'21
7419 1156 00'00'11
741C 1157 0000'CD
741F 1158
741F 1159 03 0E 00'00'21
              0D 23 77 AF
              0425'C2
742B 1160 0009'3A
742E 1161 0003'32
7431 1162
7431 1163 04 0E 00'00'21
              0D 23 77 AF
              0437'C2
743D 1164
743D 1165 00'00'21
7440 1166 02D2'CD
7443 1167
7443 1168 044C'C3
7446 1169
7446 1170
7446 1171

```

N 2

Fiche 1 Frame N2

Sequence 26

IN PREPARATION FOR USING THE CON_ERROR ROUTINE.

TESTC_ERROR SAME FOR TEST C

```

*****
TESTD_ERROR:  MVI    A,1           ;ERROR NUMBER 1.
               STA    CON_ERR_NUM
TESTC_ERROR:  LXI    H,EXP_DAT      ;MOVE EXPECTED DATA TO CON_EXP_DAT.
               LXI    D,CON_EXP_DAT
               CALL   MOVER_4
               LXI    H,REC_DAT      ;MOVE RECEIVED DATA TO CON_REC_DAT.
               LXI    D,CON_REC_DAT
               CALL   MOVER_4
               SET    NA_OTHER       ;NO ADDITIONAL DATA.
               ZERO   CON_MSK_DAT,4  ;ERROR MASK INCLUDES ALL BITS.

               JMP    REPORT_CPU_ERROR ;REPORT ERROR ON CPU BOARD

```

TESTE_F_ERROR THIS ROUTINE IS USED BY TESTS E AND F TO DEFINE THE ERROR VARIABLES IN PREPARATION FOR USING THE CON_ERROR ROUTINE.

```

*****
TESTE_F_ERROR: LXI    H,EXP_DAT      ;MOVE EXPECTED DATA TO CON_EXP_DAT.
               LXI    D,CON_EXP_DAT
               CALL   MOVER_4
               LXI    H,DATA0        ;MOVE RECEIVED DATA TO CON_REC_DAT.
               LXI    D,CON_REC_DAT
               CALL   MOVER_4
               ZERO   CON_ADD_DAT,3  ;ADDITIONAL DATA CONTAINS LS

               LDA    LS_ADDR        ; POINTER.
               STA    CON_ADD_DAT+3
               ZERO   CON_MSK_DAT,4  ;ERROR MASK INCLUDES ALL BITS.

               LXI    H,MWCS_A       ;WCS MODULE BEING TESTED.
               CALL   LOAD_MOD_DAT
               JMP    REPORT_ERROR   ;REPORT ERROR

```

```

ZZ-ENKBB-2.0    7000    4
7446 1172
7446 1173
7446 1174
7446 1175
7446 1176
7446 1177
7446 1178
7446 1179
7446 1180 00'00'21
7449 1181 02D2'CD
744C 1182
744C 1183 0000'CD
744F 1184
744F 1185 C9
7450 1186
7450 1187
7450 1188
7450 1189
7450 1190
7450 1191
7450 1192
7450 1193
7450 1194
7450 1195
7450 1196
7450 1197
7450 1198
7450 1199
7450 1200
7450 1201
7450 1202
7450 1203
7450 1204
7450 1205
7450 1206
7450 1207
7450 1208
7450 1209
7450 1210
7450 1211
7450 1212
7450 1213
7450 1214
7450 1215
7450 1216
7450 1217
7450 1218
7450 1219
7450 1220
7450 1221
7450 1222
7450 1223
7450 1224
7450 1225
7450 1226
7450 1227
7450 1228
7450 1229
7450 1230
7450 1231

```

B 3

Fiche 1 Frame B3

Sequence 27

```

: REPORT_CPU_ERROR THIS ROUTINE LOADS MODULE DATA WITH CODE FOR CPU
: BOARD AND CALLS THE ERROR REPORT ROUTINE
:
: REPORT_ERROR THIS ROUTINE CALLS THE ERROR REPORT ROUTINE ONLY
:
:*****

```

```

REPORT_CPU_ERROR:
                  LXI      H,MCPU_A      ;CPU MODULE BEING TESTED.
                  CALL     LOAD_MOD_DAT
REPORT_ERROR:     CALL     CON_ERROR      ;PRINT AN ERROR.
                  RET

```

TEST9 2901 DATA PATH TEST (WORKING REG 0 AND WR CRR TESTS) -----

DATA WILL BE SENT TO A 2901 WORKING REGISTER OVER THE 8 BIT DATA PATH AND THEN READ BACK. THE DATA SENT WILL BE ALL 1'S, ALL 0'S, ONE RIPPLED THROUGH A FIELD OF ZEROS AND A ZERO RIPPLED THROUGH A FIELD OF ONES.

TEST STEPS:

- 1) WRITE DATA TO WORKING REG.
- 2) READ DATA BACK.
- 3) PRINT ERRORS IF ANY.
- 4) REPEAT 1-3 FOR ALL PATTERNS.
- 5) REPEAT, CLOCKING LATCH WITH WR CRR
- 6) TEST FOR CLOCK LOGIC STUCK SET

ASSUMPTIONS: THE D AND Y-BUSES ARE WORKING AS TESTED IN THE PREVIOUS TEST.

```

ERROR1: 2901 DATA PATH TEST ERROR (WORKING REG 0).
ERROR2: 2901 DATA PATH TEST ERROR (WR CRR).
ERROR3: 2901 DATA PATH TEST ERROR

```

DEBUG: FOR WRITE TO REG:

- 1) CHECK 'SRC CTL H' FOR 'D.O' = 7.
- 2) CHECK 'ALU CTL H' FOR 'R.OR.S' = 3.
- 3) CHECK 'DEST CTL H' FOR 'WRITE.B.F' = 3.

FOR READ OF REG:

- 4) CHECK 'SRC CTL H' FOR 'O.A' = 4.
- 5) CHECK 'ALU CTL H' FOR 'R.OR.S' = 3.
- 6) CHECK 'DEST CTL H' FOR 'NOP' = 1.

22-ENKBB-2.0 7000 4

7450 1232
7450 1233
7450 1234
7450 1235
7450 1236
7450 1237
7450 1238
7450 1239
7450 1240
7450 1241
7450 1242 0000'CD 09 3E
OF 0000'3A
C9 045D'D2
OF 0000'3A
3D D3 0515'DA

7466 1243
7466 1244 00'0E'21
7469 1245 029C'CD
746C 1246
746C 1247 0054'3A
746F 1248
746F 1249 02DB'CD
7472 1250
7472 1251 0466'DA
7475 1252
7475 1253 00'0E'21
7478 1254 029C'CD
747B 1255
747B 1256 0059'3A
747E 1257
747E 1258 02DB'CD
7481 1259
7481 1260 0475'DA
7484 1261
7484 1262
7484 1263
7484 1264 0058'3A
7487 1265 0087'32
748A 1266
748A 1267 C5 46 00'00'21
77 08 3E
7492 1268
7492 1269 0307'CD
7495 1270
7495 1271 0087'3A
7498 1272 07
7499 1273 0087'32
749C 1274
749C 1275 35 00'00'21
70 C1 0492'C2

74A5 1276
74A5 1277
74A5 1278 005D'3A
74A8 1279 0087'32
74AB 1280
74AB 1281 C5 46 00'00'21
77 08 3E
74B3 1282
74B3 1283 0307'CD
74B6 1284

C 3

Fiche 1 Frame C3

Sequence 28

7) CHECK 'SRC CTL H,DEST CTL H,ALU CTL H' FOR CSR 22 - 14.
8) B ADDRESS 0.

ERROR 2: CHECK WRITE CRR L DECODER
ERROR 3: CHECK Y-BUS LATCH CLOCK LOGIC

TEST9: HEAD <^X09>

1\$: LXI H,MOV_CCR_WRO ;LOAD THE CSR WITH A MOVE INSTRUCTION
CALL LD_CSR_INST ; USED TO PASS DATA THROUGH THE ALU.
LDA ZERO_DAT ;LOAD THE FIRST DATA PATTERN INTO A
CALL TEST9_1_0

JC 1\$;...JUMP TO BEGINNING OF ERROR LOOP.

2\$: LXI H,MOV_CCR_WRO ;LOAD THE CSR WITH A MOVE INSTRUCTION
CALL LD_CSR_INST ; USED TO PASS DATA THROUGH THE ALU.
LDA ONE_DAT ;LOAD ALL ZEROS INTO A
CALL TEST9_1_0
JC 2\$;...JUMP TO BEGINNING OF ERROR LOOP.

LDA ONE_SHIFT+3 ;LOAD A ONE THRIUGH A ZERO FIELD
STA TEMP_A_DAT ; PATTERN INTO TEMP_A_DAT.

DO 8

CALL TEST9_SHIFT

LDA TEMP_A_DAT ;ROTATE THE TEST PATTERN TO THE LEFT
RLC ; TO SHIFT THE ONE THROUGH THE ZERO
STA TEMP_A_DAT ; FIELD.

ENDDO

LDA ZERO_SHIFT+3 ;LOAD A ZERO THROUGH A ONE FIELD
STA TEMP_A_DAT ; PATTERN INTO TEMP_A_DAT.

DO 8

CALL TEST9_SHIFT

```

22-ENKBB-2.0 7000 4
74B6 1285 0087'3A
74B9 1286 07
74BA 1287 0087'32
74BD 1288
74BD 1289 35 00'00'21
70 C1 04B3'C2
74C6 1290
74C6 1291
74C6 1292 0054'3A
74C9 1293
74C9 1294 033F'CD
74CC 1295
74CC 1296 04C6'DA
74CF 1297
74CF 1298
74CF 1299 0059'3A
74D2 1300
74D2 1301 033F'CD
74D5 1302
74D5 1303 04CF'DA
74D8 1304
74D8 1305
74D8 1306 00'0E'21
74DB 1307 029C'CD
74DE 1308
74DE 1309 0054'3A
74E1 1310
74E1 1311 CC D3
74E3 1312
74E3 1313 23 D3
74E5 1314 22 D3
74E7 1315
74E7 1316 00'00'21
74EA 1317 029C'CD
74ED 1318
74ED 1319 23 D3
74EF 1320 22 D3
74F1 1321
74F1 1322 80 DB
74F3 1323
74F3 1324 0095'32
74F6 1325
74F6 1326 00'90'21
01 0E 00'95'11
050C'CA 0000'CD
7504 1327
7504 1328
7504 1329 03 3E
7506 1330 0000'32
7509 1331
7509 1332 0380'CD
750C 1333
750C 1334
750C 1335
750C 1336 0000'3A
750F 1337 0F
7510 1338 04D8'DA
7513 1339 3C D3
7515 1340
7515 1341

```

```

*
*
*
*
*****

```

```

*****
*
*
*
*
*****

```

D 3

```

LDA  TEMP_A
RLC
STA  TEMP_A

ENDDO

```

Fiche 1 Frame D3 Sequence 29
; ROTATE THE TEST PATTERN TO THE LEFT
; TO SHIFT THE ZERO THROUGH THE ONE
; FIELD.

```

LDA  ZERO_DAT      ;LOAD ALL ZEROS INTO A
CALL TEST9_2_0
JC   3$            ;...JUMP TO BEGINNING OF ERROR LOOP

LDA  ONE_DAT       ;LOAD ALL ONES INTO A
CALL TEST9_2_0
JC   4$            ;...JUMP TO BEGINNING OF ERROR LOOP

LXI  H,MOV_CCR_WRO ;LOAD THE CSR WITH A MOVE INSTRUCTION
CALL LD_CSR_INST   ; USED TO PASS DATA THROUGH THE ALU

LDA  ZERO_DAT
OUT  WRITE          ;WRITE PATTERN TO THE D-BUS
OUT  SETSS          ;CLOCK DATA PATH
OUT  CLRSS

LXI  H,X MOV_WRWR  ;LOAD THE CSR WITH AN INSTRUCTION
CALL LD_CSR_INST   ; TO ENABLE THE WR TO BE READ

OUT  SETSS          ;CLOCK DATA PATH
OUT  CLRSS

IN   READ           ;READ THE Y-BUS CONTENTS
STA  REC_DAT        ;STORE THE Y-BUS CONTENTS IN REC_DAT
IFNEQ EXP_DAT,REC_DAT,1 ;COMPARE THE EXPECTED AND RECEIVED

; DATA

MVI  A,3            ;ERROR NUMBER 3
STA  CON_ERR_NUM

CALL TEST9_ERR2_3   ;REPORT ERROR

ENDIF

LDA  ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC   5$             ;...JUMP TO BEGINNING OF ERROR LOOP
TAIL

```

7515 1342
7515 1343
7515 1344
7515 1345
7515 1346
7515 1347
7515 1348
7515 1349
7515 1350
7515 1351
7515 1352
7515 1353
7515 1354
7515 1355
7515 1356
7515 1357
7515 1358
7515 1359
7515 1360
7515 1361
7515 1362
7515 1363
7515 1364
7515 1365
7515 1366
7515 1367
7515 1368
7515 1369
7515 1370
7515 1371
7515 1372
7515 1373
7515 1374
7515 1375
7515 1376
7515 1377
7515 1378
7515 1379
7515 1380
7515 1381
7515 1382
7515 1383
7515 1384
7515 1385
7515 1386
7515 1387
7515 1388
7515 1389
7515 1390
7515 1391
7515 1392
7515 1393
7515 1394
7515 1395
7515 1396
7515 1397
7515 1398
7515 1399
7515 1400
7515 1401

TESTA

2901 SHIFT FUNCTION TEST

A ONE WILL BE PASSED TO THE 2901 IN BIT POSITION ZERO AND SHIFTED TO THE LEFT 4 BITS THEN TO THE RIGHT 4 BITS AND READ BACK. THIS WILL THEN BE FOLLOWED BY ANOTHER WRITE TO THE 2901 IN THE SAME MANNER EXCEPT THE SHIFT WILL BE 8 BITS, THEN AGAIN WITH 12 BITS AND SO ON UP TO THE 32 BITS OF THE 2901. THIS WILL GIVE ISOLATION TO THE 4 BIT SLICE OF THE 2901 THAT MIGHT FAIL TO SHIFT.

NOTE: DATA WILL BE READ FROM THE LOWER 8 BITS TO VERIFY THAT THE DATA WAS IN FACT SHIFTED OUT OF BIT POSITION 0

TEST STEPS:

- 1) LOAD 2901 WORKING REGISTER WITH BIT 0 ON.
- 2) SHIFT LEFT 8 TIMES.
- 3) READ LOWER 8 BITS TO VERIFY DATA SHIFTED AWAY.
- 4) LOAD 2901 WORKING REGISTER WITH BIT 0 ON.
- 5) SHIFT LEFT N TIMES.
- 6) READ LOWER 4 BITS TO VERIFY DATA SHIFTED AWAY.
- 7) SHIFT RIGHT N TIMES.
- 8) READ BACK DATA.
- 9) PRINT ERROR IF ANY.
- 10) INC N BY 4.
- 11) REPEAT 4-10 UNTIL 32 BITS OF SHIFT HAVE BEEN DONE.

ASSUMPTIONS: THE DATA PATH TO AND FROM THE 2901 IS WORKING AS TESTED IN PREVIOUS TESTS.

ERROR1: DATA NOT SHIFTING AWAY.

ERROR2: DATA NOT BEING SHIFTED BACK CORRECTLY.

NOTE: ADDITIONAL DATA WILL GIVE THE SHIFT DEPTH (4,8,12...).

- DEBUG:
- 1) CHECK THAT EACH BIT SLICE CAN SHIFT. FOR AN ERROR IN SHIFTING AT ANY BIT POSITION N AN ERROR COULD OCCUR IN THE SERIAL IN LINES CONNECTING THE BIT SLICES OR THE SERIAL SHIFT OF THE PREVIOUS BIT SLICE.
 - 2) ON ROL INSTRUCTIONS
"SRC CTL H" FOR "O.B" = 3
"ALU CTL H" FOR "R.OR.S" = 3
"DEST CTL H" FOR "LSHF.RAM" = 7
 - 3) ON ROR INSTRUCTIONS
"SRC CTL H" FOR "O.B" = 3
"ALU CTL H" FOR "R.OR.S" = 3
"DEST CTL H" FOR "RSHF.RAM" = 5
 - 4) CHECK "SRC CTL H,DEST CTL H,ALU CTL H" FOR CSR 22 - 14

22-ENKBB-2.0 7000 4
7515 1402 0000'CD 0A 3E
OF 0000'3A
C9 0522'D2
OF 0000'3A
3D D3 0609'DA

752B 1403
752B 1404 01 3E
752D 1405 0000'32
7530 1406
7530 1407 08 3E
7532 1408 0008'32
7535 1409
7535 1410 0090'32 AF
7539 1411
7539 1412
7539 1413 00'0E'21
753C 1414 029C'CD
753F 1415
753F 1416 01 3E
7541 1417 CC D3
7543 1418
7543 1419 23 D3
7545 1420 22 D3
7547 1421
7547 1422 0008'3A
754A 1423 0007'32
754D 1424
754D 1425 0000'CD
7550 1426
7550 1427 00A9'CD
7553 1428
7553 1429 0007'3A
7556 1430 3D
7557 1431 0007'32
755A 1432 054D'C2
755D 1433
755D 1434 00'00'21
7560 1435 029C'CD
7563 1436
7563 1437 EC D3
7565 1438 80 DB
7567 1439 0095'32
756A 1440
756A 1441 00'90'21
01 0E 00'95'11
057B'CA 0000'CD

7578 1442
7578 1443
7578 1444 03A0'CD
757B 1445
757B 1446
757B 1447
757B 1448 0000'3A
757E 1449 OF
757F 1450 0539'DA
7582 1451
7582 1452
7582 1453 02 3E
7584 1454 0000'32
7587 1455

*
*
*
*

TESTA:

F 3

HEAD <^X0A>

Fiche 1 Frame F3

Sequence 31

MVI A,1 ;FIRST PART OF TEST A.
STA CON_ERR_NUM
MVI A,8 ;SET SHIFT_CNT_DAT TO 8.
STA SHIFT_CNT_DAT
CLEAR EXP_DAT ;EXPECTED DATA IS ZERO AS DATA
; SHOULD GET SHIFTED AWAY.
1\$: LXI H,MOV_CCR_WRO ;LOAD THE CSR WITH AN INSTRUCTION TO
CALL LD_CSR_INST ; ENABLE THE 2901 TO BE LOADED.
MVI A,1 ;LOAD A ONE INTO A.
OUT WRITE ;WRITE THE DATA TO THE D-BUS LATCH.
OUT SETSS ;LOAD THE DATA INTO THE 2901'S WRO.
OUT CLRSS
LDA SHIFT_CNT_DAT ;LOAD THE SHIFT COUNTER DATA INTO THE
STA SHIFT_CNT ; SHIFT COUNTER.
2\$: CALL SHIFT_L_2901 ;SHIFT THE DATA TO THE LEFT ONE BIT.
CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL
LDA SHIFT_CNT ;DECREMENT THE SHIFT COUNTER AND SHIFT
DCR A ; AGAIN IF APPROPRIATE.
STA SHIFT_CNT
JNZ 2\$
LXI H,X MOV_WRRW ;PREPARE TO READ THE LOW BYTE OF WRO.
CALL LD_CSR_INST
OUT YBUSRD ;LOAD THE DATA INTO THE Y-BUS LATCH.
IN READ ;READ THE Y-BUS.
STA REC_DAT ;STORE THE DATA IN REC_DAT.
IFNEQ EXP_DAT,REC_DAT,1 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.
CALL TESTA_ERROR ;PRINT AN ERROR.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1\$;...JUMP TO BEGINNING OF ERROR LOOP.
MVI A,2 ;SECOND PART OF TEST A.
STA CON_ERR_NUM


```

ZZ-ENKBB-2.0 7000 4
7587 1456 04 3E
7589 1457 0008'32
758C 1458
758C 1459 C5 46 00'00'21
77 07 3E
7594 1460
7594 1461 00'0E'21
7597 1462 029C'CD
759A 1463
759A 1464 01 3E
759C 1465 0090'32
759F 1466 CC D3
75A1 1467
75A1 1468 23 D3
75A3 1469 22 D3
75A5 1470
75A5 1471 0008'3A
75A8 1472 0007'32
75AB 1473
75AB 1474 0000'CD
75AE 1475
75AE 1476 00A9'CD
75B1 1477
75B1 1478 0007'3A
75B4 1479 3D
75B5 1480 0007'32
75B8 1481 05AB'C2
75BB 1482
75BB 1483 0008'3A
75BE 1484 0007'32
75C1 1485
75C1 1486 00'01'21
75C4 1487 0294'CD
75C7 1488
75C7 1489 0007'3A
75CA 1490 3D
75CB 1491 0007'32
75CE 1492 05C1'C2
75D1 1493
75D1 1494 00'00'21
75D4 1495 029C'CD
75D7 1496
75D7 1497 EC D3
75D9 1498 80 DB
75DB 1499 0095'32
75DE 1500
75DE 1501 00'90'21
01 0E 00'95'11
05EF'CA 0000'CD
75EC 1502
75EC 1503
75EC 1504 03A0'CD
75EF 1505
75EF 1506
75EF 1507
75EF 1508 0000'3A
75F2 1509 0F
75F3 1510 0594'DA
75F6 1511
75F6 1512 0008'3A

```

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

G 3

Fiche 1 Frame G3 Sequence 32

```

MVI A,4 ;SET SHIFT_CNT_DAT TO FOUR.
STA SHIFT_CNT_DAT
DO 7 ;7 SHIFT DEPTHS ARE USED (4,8,12...28).

LXI H,MOV_CCR_WRO ;LOAD THE CSR WITH AN INSTRUCTION TO
CALL LD_CSR_INST ; ENABLE THE 2901 TO BE LOADED.

MVI A,1 ;LOAD A ONE INTO A AND INTO EXP_DAT.
STA EXP_DAT
OUT WRITE ;WRITE THE DATA TO THE D-BUS LATCH.

OUT SETSS ;LOAD THE DATA INTO THE 2901'S WRO.
OUT CLRSS

LDA SHIFT_CNT_DAT ;LOAD THE SHIFT COUNTER DATA INTO THE
STA SHIFT_CNT ; SHIFT COUNTER.

CALL SHIFT_L_2901 ;SHIFT THE DATA TO THE LEFT ONE BIT.

CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL

LDA SHIFT_CNT ;DECREMENT THE SHIFT COUNTER AND SHIFT
DCR A ; AGAIN IF APPROPRIATE.
STA SHIFT_CNT
JNZ 4$

LDA SHIFT_CNT_DAT ;LOAD THE SHIFT COUNTER DATA INTO THE
STA SHIFT_CNT ; SHIFT COUNTER.

LXI H,X_SHIFT_R+1 ;PERFORM THE INSTRUCTION USED TO SHIFT
CALL EXEC_INST ; WR[0] TO THE RIGHT ONE PLACE.

LDA SHIFT_CNT ;DECREMENT THE SHIFT COUNTER AND SHIFT
DCR A ; AGAIN IF APPROPRIATE.
STA SHIFT_CNT
JNZ 5$

LXI H,X_MOV_WRRW ;PREPARE TO READ THE DATA BACK.
CALL LD_CSR_INST

OUT YBUSRD ;LOAD THE DATA INTO THE Y-BUS LATCH.
IN READ ;READ THE Y-BUS.
STA REC_DAT ;STORE THE DATA IN REC_DAT.

IFNEQ EXP_DAT,REC_DAT,1 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.

CALL TESTA_ERROR ;PRINT AN ERROR.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 3$ ;...JUMP TO BEGINNING OF ERROR LOOP.

LDA SHIFT_CNT_DAT ;ADD 4 TO THE VALUE OF SHIFT_CNT_DAT

```

22-ENKBB-2.0 7000 4

75F9 1513 04 C6

75FB 1514 0008'32

75FE 1515

75FE 1516 35 00'00'21

70 C1 0594'C2

7607 1517

7607 1518 3C D3

7609 1519

7609 1520

7609 1521

7609 1522

7609 1523

7609 1524

7609 1525

7609 1526

7609 1527

7609 1528

7609 1529

7609 1530

7609 1531

7609 1532

7609 1533

7609 1534

7609 1535

7609 1536

7609 1537

7609 1538

7609 1539

7609 1540

7609 1541

7609 1542

7609 1543

7609 1544

7609 1545

7609 1546

7609 1547

7609 1548

7609 1549

7609 1550

7609 1551

7609 1552

7609 1553

7609 1554

7609 1555

7609 1556

7609 1557

7609 1558

7609 1559

7609 1560

7609 1561

7609 1562

7609 1563

7609 1564

7609 1565

7609 1566

7609 1567

7609 1568

7609 1569

7609 1570

7609 1571

H 3

Fiche 1 Frame H3

Sequence 33

ADI 4

STA

SHIFT_CNT_DAT

; TO SHIFT THE DATA FOUR BITS DEEPER
; INTO THE 2901 ON THE NEXT PASS.

ENDDO

TAIL

TESTB

WORKING REGISTER CLEAR TEST

THIS TEST WILL LOAD A WORKING REGISTER WITH EACH OF THE
STANDARD PATTERNS OF ONE RIPPLED THROUGH A FIELD OF ZEROS AND A
ZERO RIPPLED THROUGH A FIELD OF ONES. THE REGISTER WILL THEN BE
CLEARED AND READ TO BE SURE IT IS CLEARED. BECAUSE THE READS AND
WRITES OF WRO USED SUPPORT MICROCODE, THE FIRST PART OF THIS TEST
PERFORMS A FAST WRITE WRO VERIFICATION. THE SUPPORT CODE IS SINGLE
STEPPED THROUGH AND THE VALUE OF THE UPC IS CHECKED AFTER EACH
INSTRUCTION.

TEST STEPS:

- 1) STEP THROUGH SUPPORT MICROCODE.
- 2) COMPARE EXPECTED AND ACTUAL UPC VALUE AFTER EACH STEP.
- 3) PRINT ERROR IF ANY.
- 4) EXECUTE A CLEAR WRO AND THEN A COMPLEMENT WRO INSTRUCTION.
READ WRO AND CHECK FOR ALL ONES.
- 5) WRITE ALL ONES TO WRO AND THEN READ WRO AND CHECK FOR ALL
ONES.
- 6) LOAD WR WITH DATA PATTERN.
- 7) PUT CLR WR INST IN CSR AND EXECUTE.
- 8) READ THE WORKING REGISTER TO PROVE IT WAS CLEARED.
- 9) PRINT ERROR IF ANY.
- 10) REPEAT 6-9 FOR ALL DATA PATTERNS.

ERROR1: VERIFY FAST WRITE FAILURE. ADDITIONAL DATA WILL SPECIFY
WHETHER CPU ATTN IS CLEAR: ADD DAT=0, OR SET: ADD DAT=1.

ERROR2: WORKING REGISTER DOES NOT CONTAIN ALL ONES AFTER CLEAR,
COMPLEMENT INSTRUCTION SEQUENCE.

ERROR3: WORKING REGISTER DOES NOT CONTAIN ALL ONES AFTER WRITING
ALL ONES TO IT.

ERROR4: CLEAR WORKING REGISTER TEST FAILURE. FOR THIS ERROR,
ADDITIONAL DATA WILL CONTAIN THE DATA SENT TO WRO.

DEBUG: 1) CLR WRC] USES 'WR.XOR.WR'
'SRC CTL H' FOR 'A.B' = 1
'ALU CTL H' FOR 'R.XOR.S' = 6
'DEST CTL H' FOR 'WRITE B.F' = 3
2) CHECK 'SRC CTL H,DEST CTL H,ALU CTL H' FOR CSR 22 - 14

7609	1572	
7609	1573	0000'CD 0B 3E OF 0000'3A C9 0616'D2 OF 0000'3A 3D D3 083E'DA
761F	1574	
761F	1575	00B9'CD
7622	1576	
7622	1577	
7622	1578	
7622	1579	01 3E
7624	1580	0000'32
7627	1581	
7627	1582	04 0E 00'00'21 0D 23 77 AF 062D'C2
7633	1583	
7633	1584	FF FF 21
7636	1585	0000'22
7639	1586	02 0E 00'02'21 0D 23 77 AF 063F'C2
7645	1587	
7645	1588	00 00 21
7648	1589	0000'22
764B	1590	0000'CD
764E	1591	
764E	1592	0000'CD
7651	1593	
7651	1594	00'44'21
7654	1595	0000'22
7657	1596	
7657	1597	AB D3
7659	1598	
7659	1599	C5 46 00'00'21 77 05 3E
7661	1600	
7661	1601	00A9'CD
7664	1602	
7664	1603	0000'CD
7667	1604	011B'CD
766A	1605	
766A	1606	
766A	1607	
766A	1608	00'92'21 02 0E 00'97'11 067B'CA 0000'CD
7678	1609	
7678	1610	
7678	1611	03CD'CD
767B	1612	
767B	1613	
767B	1614	
767B	1615	35 00'00'21 70 C1 0661'C2
7684	1616	
7684	1617	0000'3A
7687	1618	OF
7688	1619	0645'DA

TESTB:

```

HEAD      <^X0B>

CALL      LD_SUPPORT      ;LOAD SUPPORT CODE FOR FAST READS AND
                        ; WRITES OF WRO.

MVI       A,1             ;FIRST PART OF TEST B.
STA       CON_ERR_NUM

ZERO      CON_ADD_DAT,4   ;CLEAR ADDITIONAL DATA.

LXI       H,<^XFFFF>      ;LAST TWO BYTES OF EXPECTED AND
SHLD      CON_MSK_DAT     ; RECEIVED DATA ARE OF INTEREST.
ZERO      CON_MSK_DAT+2,2

LXI       H,0             ;LOAD A ZERO INTO THE UPC.
SHLD      UPC_VALUE
CALL      LOAD_UPC

CALL      NOP_CSR         ;LOAD THE CSR WITH A NOP.

LXI       H,VER_WRITE_TAB ;POINT TO THE TOP OF THE VERIFY
SHLD      WRITE_TAB_PNT   ; WRITE TABLE.

OUT       CLRATN          ;CLEAR CPU ATTENTION.

DO        5

CALL      CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL

CALL      MICRO_STEP_CPU  ;EXECUTE ONE MICROINSTRUCTION.
CALL      SET_COMPARE     ;READ THE UPC AND PUT IN REC_DAT.
                        ;ALSO, PUT THE CORRESPONDING WRITE
                        ; TABLE ENTRY IN EXP_DAT.

IFNEQ     EXP_DAT+2,REC_DAT+2,2 ;COMPARE THE EXPECTED AND
                                ; RECEIVED DATA.

CALL      TESTB_ERROR     ;PRINT AN ERROR MESSAGE.

ENDIF

ENDDO

LDA       ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC        1$              ;...JUMP TO BEGINNING OF ERROR LOOP.

```

ZZ-ENKBB-2.0 7000 4

```

768B 1620
768B 1621 00'4E'21
768E 1622 0000'22
7691 1623
7691 1624 AA D3
7693 1625
7693 1626 C5 46 00'00'21
          77 03 3E
          *****
          *
          *
769B 1627
769B 1628 0CA9'CD
769E 1629
769E 1630 0000'CD
76A1 1631 011B'CD
76A4 1632
76A4 1633
76A4 1634
76A4 1635 00'92'21
          02 0E 00'97'11
          06BA'CA 0000'CD
          *****
          *
          *
76B2 1636
76B2 1637
76B2 1638 0003'32 3C AF
76B7 1639
76B7 1640
76B7 1641 03CD'CD
76BA 1642
76BA 1643
76BA 1644
76BA 1645 35 00'00'21
          70 C1 069B'C2
          *****
          *
          *
76C3 1646
76C3 1647 0000'3A
76C6 1648 0F
76C7 1649 068B'DA
76CA 1650
76CA 1651
76CA 1652
76CA 1653 02 3E
76CC 1654 0000'32
76CF 1655
76CF 1656 04 0E 00'00'21
          0D 23 77 AF
          06D5'C2
          *****
          *
          *
76DB 1657
76DB 1658
76DB 1659 00'59'21
76DE 1660 00'90'11
76E1 1661 0000'CD
76E4 1662
76E4 1663 00'01'21
76E7 1664 0294'CD
76EA 1665
76EA 1666 00'00'21
76ED 1667 0294'CD
76F0 1668
76F0 1669 01B9'CD
76F3 1670
76F3 1671 00'00'21
76F6 1672 00'95'11
76F9 1673 0000'CD

```

2\$:

3\$:

J 3

Fiche 1 Frame J3

Sequence 35

```

LXI  H,VER_WRITE TAB+10 ;POINT TO THE PROPER TABLE
SHLD WRITE_TAB_PNT      ; ENTRY TO COMPARE THE UPC.

OUT  SETATN              ;SET CPU ATTENTION.

DO   3

CALL  CHECK_RD           ;CALL BOOT CODE IF UNDER RD CONTROL

CALL  MICRO_STEP_CPU     ;EXECUTE ONE MICROINSTRUCTION.
CALL  SET_COMPARE        ;READ THE UPC AND PUT IN REC_DAT.
                          ;ALSO, PUT THE CORRESPONDING WRITE
                          ; TABLE ENTRY IN EXP_DAT.

IFNEQ EXP_DAT+2,REC_DAT+2,2 ;COMPARE THE EXPECTED AND
                          ; RECEIVED DATA.

SET   CON_ADD_DAT+3      ;ADDITIONAL DATA INDICATES CPU ATTN
                          ; WAS SET WHEN ERROR OCCURRED.

CALL  TESTB_ERROR        ;PRINT AN ERROR MESSAGE.

ENDIF

ENDDO

LDA   ERROR_CON          ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC    2$                 ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI   A,2
STA   CON_ERR_NUM        ;SECOND PART OF TESTB.

ZERO  CON_MSK_DAT,4      ;ALL OF EXPECTED AND RECEIVED
                          ; DATA IS OF INTEREST.

LXI   H,ONE_DAT          ;LOAD ALL ONES INTO EXP_DAT.
LXI   D,EXP_DAT
CALL  MOVER_4

LXI   H,X_CLR_WRO+1      ;CLEAR WRO.
CALL  EXEC_INST

LXI   H,X_COM_WRO        ;COMPLEMENT WRO.
CALL  EXEC_INST

CALL  FAST_READ_WRO      ;READ BACK THE CONTENTS OF WRO.

LXI   H,DATA0            ;MOVE THE DATA IN DATA0 TO REC_DAT.
LXI   D,REC_DAT
CALL  MOVER_4

```

ZZ-ENKBB-2.0 7000 4

```
76FC 1674
76FC 1675 00'90'21 *****
          04 0E 00'95'11 *
          0712'CA 0000'CD *
770A 1676 *
770A 1677 *
770A 1678 0000'32 3C AF *
770F 1679 *
770F 1680 03CD'CD *
7712 1681 *
7712 1682 *****
7712 1683 *
7712 1684 00A9'CD *
7715 1685 *
7715 1686 0000'3A *
7718 1687 0F *
7719 1688 06E4'DA *
771C 1689 *
771C 1690 *
771C 1691 *
771C 1692 *
771C 1693 03 3E *
771E 1694 0000'32 *
7721 1695 *
7721 1696 00'59'21 *
7724 1697 00'90'11 *
7727 1698 0000'CD *
772A 1699 *
772A 1700 00'59'21 4$:
772D 1701 00'00'11 *
7730 1702 0000'CD *
7733 1703 *
7733 1704 0172'CD *
7736 1705 *
7736 1706 01B9'CD *
7739 1707 *
7739 1708 00'00'21 *
773C 1709 00'95'11 *
773F 1710 0000'CD *
7742 1711 *
7742 1712 00'90'21 *****
          04 0E 00'95'11 *
          0758'CA 0000'CD *
7750 1713 *
7750 1714 *
7750 1715 0000'32 3C AF *
7755 1716 *
7755 1717 03CD'CD *
7758 1718 *
7758 1719 *****
7758 1720 *
7758 1721 00A9'CD *
775B 1722 *
775B 1723 0000'3A *
775E 1724 0F *
775F 1725 072A'DA *
7762 1726 *
7762 1727 *
7762 1728 *
7762 1729 *
```

K 3

Fiche 1 Frame K3

Sequence 36

```
IFNEQ EXP_DAT,REC_DAT,4 ;COMPARE THE EXPECTED AND
                                ; RECEIVED DATA.
SET NA_OTHER ;NO ADDITIONAL DATA.
CALL TESTB_ERROR ;PRINT AN ERROR.
ENDIF
CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 3$ ;....JUMP TO BEGINNING OF ERROR LOOP.

MVI A,3 ;THIRD PART OF TESTB.
STA CON_ERR_NUM

LXI H,ONE_DAT ;LOAD ALL ONE INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_4

LXI H,ONE_DAT ;LOAD ALL ONES INTO DATA0.
LXI D,DATA0
CALL MOVER_4

CALL FAST_WRITE_WRO ;WRITE THIS TEST PATTERN INTO WRO.
CALL FAST_READ_WRO ;READ BACK THE CONTENTS OF WRO.

LXI H,DATA0 ;MOVE THE DATA IN DATA0 TO REC_DAT.
LXI D,REC_DAT
CALL MOVER_4

IFNEQ EXP_DAT,REC_DAT,4 ;COMPARE THE EXPECTED AND
                                ; RECEIVED DATA.
SET NA_OTHER ;NO ADDITIONAL DATA.
CALL TESTB_ERROR ;PRINT AN ERROR.
ENDIF
CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 4$ ;....JUMP TO BEGINNING OF ERROR LOOP.
```

```

22-ENKBB-2.0 7000 4
7762 1730 04 3E
7764 1731 0000'32
7767 1732
7767 1733 04 0E 00'90'21
          0D 23 77 AF
          076D'C2
7773 1734
7773 1735 00'55'21
7776 1736 00'8B'11
7779 1737 0C00'CD
777C 1738
777C 1739 C5 46 00'00'21
          77 20 3E
7784 1740
7784 1741 00'8B'21
7787 1742 00'00'11
778A 1743 0000'CD
778D 1744
778D 1745 0172'CD
7790 1746
7790 1747 00'01'21
7793 1748 0294'CD
7796 1749
7796 1750 01B9'CD
7799 1751
7799 1752 00'00'21
779C 1753 00'95'11
779F 1754 0000'CD
77A2 1755
77A2 1756 00'90'21
          04 0E 00'95'11
          07BC'CA 0000'CD
77B0 1757
77B0 1758
77B0 1759 00'8B'21
77B3 1760 00'00'11
77B6 1761 0000'CD
77B9 1762
77B9 1763 03CD'CD
77BC 1764
77BC 1765
77BC 1766
77BC 1767 0000'3A
77BF 1768 0F
77C0 1769 0784'DA
77C3 1770
77C3 1771 B7
77C4 1772
77C4 1773 00'8A'21
77C7 1774 0000'22
77CA 1775 0000'CD
77CD 1776
77CD 1777 35 00'00'21
          70 C1 0784'C2
77D6 1778
77D6 1779 00A9'CD
77D9 1780
77D9 1781
77D9 1782
77D9 1783

```

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

5\$:

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

L 3

Fiche 1 Frame L3 Sequence 37

```

MVI A,4
STA CON_ERR_NUM ;FOURTH PART OF TESTB.

ZERO EXP_DAT,4 ;EXPECTED DATA IS ALL ZEROS.

LXI H,ONE_SHIFT ;LOAD A ONE THROUGH A ZERO FIELD
LXI D,TEMPB_DAT ;PATTERN INTO TEMPB_DAT.
CALL MOVER_4

DO 32 ;32 DIFFERENT TEST PATTERNS.

LXI H,TEMPB_DAT ;MOVE THE PATTERN TO DATA0.
LXI D,DATA0
CALL MOVER_4

CALL FAST_WRITE_WRO ;WRITE THE PATTERN TO WRO

LXI H,X_CLR_WRO+1 ;CLEAR WRO.
CALL EXEC_INST

CALL FAST_READ_WRO ;READ THE DATA FROM WRO.

LXI H,DATA0 ;MOVE THE RETURNED DATA TO REC_DAT.
LXI D,REC_DAT
CALL MOVER_4

IFNEQ EXP_DAT,REC_DAT,4 ;COMPARE THE EXPECTED AND

; RECEIVED DATA.

LXI H,TEMPB_DAT ;ADDITIONAL DATA CONTAINS DATA
LXI D,CON_ADD_DAT ;SENT TO WRO.
CALL MOVER_4

CALL TESTB_ERROR ;PRINT AN ERROR.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 5$ ;...JUMP TO BEGINNING OF ERROR LOOP.

ORA A ;CLEAR THE CARRY.

LXI H,TEMPB_SHIFT ;SHIFT THE ONE THROUGH THE ZERO FIELD.
SHLD SHIFT_PNT
CALL SHIFTER

ENDDO

CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL

```

```

ZZ-ENKBB-2.0      7000      4
77D9 1784 00'5A'21
77DC 1785 00'8B'11
77DF 1786 0000'CD
77E2 1787
77E2 1788 C5 46 00'00'21
          77 20 3E
77EA 1789
77EA 1790 00'8B'21
77ED 1791 00'00'11
77F0 1792 0C00'CD
77F3 1793
77F3 1794 0172'CD
77F6 1795
77F6 1796 00'01'21
77F9 1797 0294'CD
77FC 1798
77FC 1799 0189'CD
77FF 1800
77FF 1801 00'00'21
7802 1802 00'95'11
7805 1803 0000'CD
7808 1804
7808 1805 00'90'21
          04 0E 00'95'11
          0822'CA 0000'CD
7816 1806
7816 1807
7816 1808 00'8B'21
7819 1809 00'00'11
781C 1810 0000'CD
781F 1811
781F 1812 03CD'CD
7822 1813
7822 1814
7822 1815
7822 1816 0000'3A
7825 1817 0F
7826 1818 07EA'DA
7829 1819
7829 1820 37
782A 1821
782A 1822 00'8A'21
782D 1823 0000'22
7830 1824 0000'CD
7833 1825
7833 1826 35 00'00'21
          70 C1 07EA'C2
783C 1827
783C 1828 3C D3
783E 1829
783E 1830
783E 1831
783E 1832
783E 1833
783E 1834
783E 1835
783E 1836
783E 1837
783E 1838
783E 1839

```

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

6\$:

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

M 3

Fiche 1 Frame M3 Sequence 38

```

LXI H,ZERO SHIFT ;LOAD A ZERO THROUGH A ONE FIELD
LXI D,TEMPB_DAT ; PATTERN INTO TEMPB_DAT.
CALL MOVER_4

```

DO 32

```

LXI H,TEMPB_DAT ;LOAD THE PATTERN INTO DATA0.
LXI D,DATA0
CALL MOVER_4

```

CALL FAST_WRITE_WRO ;WRITE THE PATTERN TO WRO

```

LXI H,X CLR WRO+1 ;CLEAR WRO.
CALL EXEC_INST

```

CALL FAST_READ_WRO ;READ THE DATA FROM WRO.

```

LXI H,DATA0 ;MOVE THE RETURNED DATA TO REC_DAT.
LXI D,REC_DAT
CALL MOVER_4

```

IFNEQ EXP_DAT,REC_DAT,4 ;COMPARE THE EXPECTED AND

; RECEIVED DATA.

```

LXI H,TEMPB_DAT ;ADDITIONAL DATA CONTAINS DATA
LXI D,CON_ADD_DAT ; SENT TO WRO.
CALL MOVER_4

```

CALL TESTB_ERROR ;PRINT AN ERROR.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...

```

RRC
JC 6$ ;...JUMP TO BEGINNING OF ERROR LOOP.

```

STC ;SET THE CARRY.

```

LXI H,TEMPB_SHIFT ;SHIFT THE ZERO THROUGH THE ONE STRING.
SHLD SHIFT_PNT
CALL SHIFTER

```

ENDDO

TAIL

TESTC

WORKING REGISTER TEST (WORKING REG 1)

A STANDARD TEST PATTERN OF RIPPLING A ZERO THROUGH A
FIELD OF ONES AND A ONE THROUGH A FIELD OF ZEROS WILL BE USED

```

ZZ-ENKBB-2.0      7000      4
783E 1840
783E 1841
783E 1842
783E 1843
783E 1844
783E 1845
783E 1846
783E 1847
783E 1848
783E 1849
783E 1850
783E 1851
783E 1852
783E 1853
783E 1854
783E 1855
783E 1856
783E 1857
783E 1858
783E 1859
783E 1860
783E 1861
783E 1862
783E 1863
783E 1864
783E 1865
783E 1866
783E 1867
783E 1868
783E 1869
783E 1870
783E 1871
783E 1872
783E 1873
783E 1874
783E 1875
783E 1876
783E 1877
783E 1878
783E 1879
783E 1880
783E 1881
783E 1882
783E 1883
783E 1884
783E 1885 0000'CD 0C 3E
              OF 0000'3A
              C9 084B'D2
              OF 0000'3A
              3D D3 0A19'DA
7854 1886
7854 1887 00B9'CD
7857 1888
7857 1889
7857 1890 01 3E
7859 1891 0000'32
785C 1892
785C 1893 00'54'21
785F 1894 00'90'11
7862 1895 0000'CD

```

N 3

Fiche 1 Frame N3

Sequence 39

TO TEST THIS WORKING REGISTER FOR FUTURE USE. WRO WILL ALSO BE
WRITTEN TO ASSURE THAT DUAL ADDRESSING DOES NOT OCCUR

TEST STEPS:

- 1) WRITE DATA TO WORKING REG 1.
- 2) READ DATA BACK.
- 3) PRINT ERRORS IF ANY.
- 4) REPEAT 1-3 FOR ALL PATTERNS.
- 5) WRITE WRO WITH 1'S.
- 6) WRITE WR1 WITH 0'S.
- 7) READ 1'S IN WRO.
- 8) READ 0'S IN WR1.
- 9) PRINT ERRORS IF ANY.

ASSUMPTIONS: THE DATA PATH TO AND FROM THE 2901 IS WORKING
AS TESTED IN PREVIOUS TESTS.

ERROR1: 2901 WORKING REGISTER TEST FAILURE.
ERROR2: DUAL ADDRESSING ERROR.

DEBUG: FOR WRITE TO REG:

- 1) CHECK 'SRC CTL H' FOR 'D.O' = 7.
- 2) CHECK 'ALU CTL H' FOR 'R.OR.S' = 3.
- 3) CHECK 'DEST CTL H' FOR 'WRITE.B.F' = 3.

FOR READ OF REG:

- 4) CHECK 'SRC CTL H' FOR 'O.A' = 4.
- 5) CHECK 'ALU CTL H' FOR 'R.OR.S' = 3.
- 6) CHECK 'DEST CTL H' FOR 'NOP' = 1.
- 7) CHECK 'SRC CTL H, DEST CTL H, ALU CTL H' FOR CSR 22 - 14.
- 8) SINCE THE PATH TO AND FROM THE 2901 HAS ALREADY
BEEN TESTED IT IS PROBABLY THE 2901 ITSELF OR
DUAL ADDRESSING.
- 9) CHECK TAT B ADDRESS BIT 0 CHANGES.

TESTC: HEAD <^XOC>

```

CALL LD_SUPPORT      ;LOAD SUPPORT CODE FOR FAST READS AND
                     ; WRITES OF WRO.

MVI A,1              ;FIRST PART OF TEST C.
STA CON_ERR_NUM

LXI H,ZERO_DAT       ;LOAD ALL ZEROS INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_4

```



```

ZZ-ENKBB-2.0  7000  4
7865 1896
7865 1897 00'54'21
7868 1898 00'00'11
7868 1899 0000'CD
786E 1900
786E 1901 0205'CD
7871 1902
7871 1903 023E'CD
7874 1904
7874 1905 00'00'21
7877 1906 00'95'11
787A 1907 0000'CD
787D 1908
787D 1909 00'90'21
          04 0E 00'95'11
          088E'CA 0000'CD

788B 1910
788B 1911
788B 1912 03E7'CD
788E 1913
788E 1914
788E 1915
788E 1916 0000'3A
7891 1917 0F
7892 1918 0865'DA
7895 1919
7895 1920
7895 1921
7895 1922 00'59'21
7898 1923 00'90'11
789B 1924 0000'CD
789E 1925
789E 1926 00'59'21
78A1 1927 00'00'11
78A4 1928 0000'CD
78A7 1929
78A7 1930 0205'CD
78AA 1931
78AA 1932 023E'CD
78AD 1933
78AD 1934 00'00'21
78B0 1935 00'95'11
78B3 1936 0000'CD
78B6 1937
78B6 1938 00'90'21
          04 0E 00'95'11
          08C7'CA 0000'CD

78C4 1939
78C4 1940
78C4 1941 03E7'CD
78C7 1942
78C7 1943
78C7 1944
78C7 1945 0000'3A
78CA 1946 0F
78CB 1947 089E'DA
78CE 1948
78CE 1949
78CE 1950
78CE 1951 00'55'21

```

```

*****
*
*
*
*
*
*****

```

```

*****
*
*
*
*
*
*****

```

1\$:

2\$:

B 4

Fiche 1 Frame B4

Sequence 40

```

LXI  H,ZERO_DAT      ;LOAD ALL ZEROS INTO DATA0.
LXI  D,DATA0
CALL  MOVER_4

CALL  WRITE_WR1_32    ;WRITE PATTERN TO WR[1].

CALL  READ_WR1_32     ;READ DATA BACK FROM WR[1].

LXI  H,DATA0          ;MOVE THE RECEIVED DATA TO REC_DAT.
LXI  D,REC_DAT
CALL  MOVER_4

IFNEQ EXP_DAT,REC_DAT,4      ;COMPARE THE EXPECTED AND

                                ; RECEIVED DATA.

CALL  TESTC_ERROR      ;PRINT AN ERROR MESSAGE.

ENDIF

LDA  ERROR_CON         ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC   1$                ;...JUMP TO BEGINNING OF ERROR LOOP.

LXI  H,ONE_DAT         ;LOAD ALL ONES INTO EXP_DAT.
LXI  D,EXP_DAT
CALL  MOVER_4

LXI  H,ONE_DAT         ;LOAD ALL ONES INTO DATA0.
LXI  D,DATA0
CALL  MOVER_4

CALL  WRITE_WR1_32    ;WRITE PATTERN TO WR[1].

CALL  READ_WR1_32     ;READ DATA BACK FROM WR[1].

LXI  H,DATA0          ;MOVE RECEIVED DATA TO REC_DAT.
LXI  D,REC_DAT
CALL  MOVER_4

IFNEQ EXP_DAT,REC_DAT,4      ;COMPARE THE EXPECTED AND

                                ; RECEIVED DATA.

CALL  TESTC_ERROR      ;PRINT AN ERROR MESSAGE.

ENDIF

LDA  ERROR_CON         ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC   2$                ;...JUMP TO BEGINNING OF ERROR LOOP.

LXI  H,ONE_SHIFT      ;LOAD ONE THROUGH A ZERO FIELD

```

```

ZZ-ENKBB-2.0 7000 4
78D1 1952 00'8B'11
78D4 1953 0000'CD
78D7 1954
78D7 1955 C5 46 00'00'21
77 20 3E
78DF 1956
78DF 1957 00'8B'21
78E2 1958 00'90'11
78E5 1959 0000'CD
78E8 1960
78E8 1961 00'8B'21
78EB 1962 00'00'11
78EE 1963 0000'CD
78F1 1964
78F1 1965 0205'CD
78F4 1966
78F4 1967 023E'CD
78F7 1968
78F7 1969 00'00'21
78FA 1970 00'95'11
78FD 1971 0000'CD
7900 1972
7900 1973 00'90'21
04 0E 00'95'11
0911'CA 0000'CD
790E 1974
790E 1975
790E 1976 03E7'CD
7911 1977
7911 1978
7911 1979
7911 1980 0000'3A
7914 1981 0F
7915 1982 08E8'DA
7918 1983
7918 1984 B7
7919 1985
7919 1986 00'8A'21
791C 1987 0000'22
791F 1988 0000'CD
7922 1989
7922 1990 35 00'00'21
70 C1 08DF'C2
792B 1991
792B 1992
792B 1993
792B 1994 00'5A'21
792E 1995 00'8B'11
7931 1996 0000'CD
7934 1997
7934 1998 C5 46 00'00'21
77 20 3E
793C 1999
793C 2000 00'8B'21
793F 2001 00'90'11
7942 2002 0000'CD
7945 2003
7945 2004 00'8B'21
7948 2005 00'00'11
794B 2006 0000'CD

```

```

*****
*

```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

3$:

```

```

*****

```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

*
```

```

4$:

```

```

C 4

```

```

LXI D,TEMPB_DAT ; FICHE 1 Frame C4 Sequence 41
CALL MOVER_4 ; PATTERN INTO TEMPB_DAT.

DO 32

LXI H,TEMPB_DAT ;LOAD TEST PATTERN INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_4

LXI H,TEMPB_DAT ;LOAD TEST PATTERN INTO DATA0.
LXI D,DATA0
CALL MOVER_4

CALL WRITE_WR1_32 ;WRITE PATTERN TO WR[1].

CALL READ_WR1_32 ;READ DATA BACK FROM WR[1].

LXI H,DATA0 ;MOVE RECEIVED DATA TO REC_DAT.
LXI D,REC_DAT
CALL MOVER_4

IFNEQ EXP_DAT,REC_DAT,4 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.

CALL TESTC_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 3$ ;...JUMP TO BEGINNING OF ERROR LOOP.

ORA A ;CLEAR THE CARRY.

LXI H,TEMPB_SHIFT ;SHIFT THE ONE THROUGH THE ZERO FIELD.
SHLD SHIFT_PNT
CALL SHIFTER

ENDDO

LXI H,ZERO_SHIFT ;LOAD ZERO THROUGH A ONE FIELD
LXI D,TEMPB_DAT ; PATTERN INTO TEMPB_DAT.
CALL MOVER_4

DO 32

LXI H,TEMPB_DAT ;LOAD TEST PATTERN INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_4

LXI H,TEMPB_DAT ;LOAD TEST PATTERN INTO DATA0.
LXI D,DATA0
CALL MOVER_4

```

794E	2007	
794E	2008	0205'CD
7951	2009	
7951	2010	023E'CD
7954	2011	
7954	2012	00'00'21
7957	2013	00'95'11
795A	2014	0000'CD
795D	2015	
795D	2016	0C'90'21 04 0E 00'95'11 096E'CA 0000'CD
796B	2017	
796B	2018	
796B	2019	03E7'CD
796E	2020	
796E	2021	
796E	2022	
796E	2023	0000'3A
7971	2024	0F
7972	2025	0945'DA
7975	2026	
7975	2027	37
7976	2028	
7976	2029	00'8A'21
7979	2030	0000'22
797C	2031	0000'CD
797F	2032	
797F	2033	35 00'00'21 70 C1 093C'C2
7988	2034	
7988	2035	
7988	2036	02 3E
798A	2037	0000'32
798D	2038	
798D	2039	00'59'21
7990	2040	00'00'11
7993	2041	0000'CD
7996	2042	
7996	2043	0172'CD
7999	2044	
7999	2045	00'54'21
799C	2046	00'00'11
799F	2047	0000'CD
79A2	2048	
79A2	2049	0205'CD
79A5	2050	
79A5	2051	01B9'CD
79A8	2052	
79A8	2053	00'59'21
79AB	2054	00'90'11
79AE	2055	0000'CD
79B1	2056	
79B1	2057	00'00'21
79B4	2058	00'95'11
79B7	2059	0000'CD
79BA	2060	
79BA	2061	00'90'21 04 0E 00'95'11 09CB'CA 0000'CD

```

CALL      WRITE_WR1_32      ;WRITE PATTERN TO WR[1].
CALL      READ_WR1_32       ;READ DATA BACK FROM WR[1].
LXI       H,DATA0           ;PUT RECEIVED DATA INTO REC_DAT.
LXI       D,REC_DAT
CALL      MOVER_4

IFNEQ     EXP_DAT,REC_DAT,4      ;COMPARE THE EXPECTED AND

                                   ; RECEIVED DATA.

CALL      TESTC_ERROR       ;PRINT AN ERROR MESSAGE.

ENDIF

LDA       ERROR_CON         ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC        4$                ;...JUMP TO BEGINNING OF ERROR LOOP.

STC                                     ;SET THE CARRY.

LXI       H,TEMPB_SHIFT     ;SHIFT THE ZERO THROUGH THE ONE FIELD.
SHLD      SHIFT_PNT
CALL      SHIFTER

ENDDO

MVI       A,2               ;SECOND PART OF TEST C.
STA       CON_ERR_NUM

LXI       H,ONE_DAT         ;LOAD ALL ONES INTO DATA0.
LXI       D,DATA0
CALL      MOVER_4

CALL      FAST_WRITE_WRO    ;WRITE ONES TO WRO.

LXI       H,ZERO_DAT        ;LOAD ALL ZEROS INTO DATA0.
LXI       D,DATA0
CALL      MOVER_4

CALL      WRITE_WR1_32      ;WRITE ZEROS TO WR1.

CALL      FAST_READ_WRO     ;READ WRO.

LXI       H,ONE_DAT         ;MOVE ALL ONES INTO EXP_DAT.
LXI       D,EXP_DAT
CALL      MOVER_4

LXI       H,DATA0           ;MOVE RECEIVED DATA TO REC_DAT.
LXI       D,REC_DAT
CALL      MOVER_4

IFNEQ     EXP_DAT,REC_DAT,4      ;COMPARE THE EXPECTED AND

```

```

MVI    A,2                ;SECOND PART OF TEST C.
STA    CON_ERR_NUM

LXI    H,ONE_DAT          ;LOAD ALL ONES INTO DATA0.
LXI    D,DATA0
CALL   MOVER_4

CALL   FAST_WRITE_WRO     ;WRITE ONES TO WRO.

LXI    H,ZERO_DAT         ;LOAD ALL ZEROS INTO DATA0.
LXI    D,DATA0
CALL   MOVER_4

CALL   WRITE_WR1_32       ;WRITE ZEROS TO WR1.

CALL   FAST_READ_WRO      ;READ WRO.

LXI    H,ONE_DAT          ;MOVE ALL ONES INTO EXP_DAT.
LXI    D,EXP_DAT
CALL   MOVER_4

LXI    H,DATA0            ;MOVE RECEIVED DATA TO REC_DAT.
LXI    D,REC_DAT
CALL   MOVER_4

IFNEQ  EXP_DAT,REC_DAT,4   ;COMPARE THE EXPECTED AND

```

ZZ-ENKBB-2.0 7000 4

79C8 2062
79C8 2063
79C8 2064 03E7'CD
79CB 2065
79CB 2066
79CB 2067
79CB 2068 0000'3A
79CE 2069 0F
79CF 2070 098D'DA
79D2 2071
79D2 2072 00'59'21
79D5 2073 00'00'11
79D8 2074 0000'CD
79DB 2075
79DB 2076 0172'CD
79DE 2077
79DE 2078 00'54'21
79E1 2079 00'00'11
79E4 2080 0000'CD
79E7 2081
79E7 2082 0205'CD
79EA 2083
79EA 2084 023E'CD
79ED 2085
79ED 2086 00'54'21
79F0 2087 00'90'11
79F3 2088 0000'CD
79F6 2089
79F6 2090 00'00'21
79F9 2091 00'95'11
79FC 2092 0000'CD
79FF 2093
79FF 2094 00'90'21
04 0E 00'95'11
0A10'CA 0000'CD

7A0D 2095
7A0D 2096
7A0D 2097 03E7'CD
7A10 2098
7A10 2099
7A10 2100
7A10 2101 0000'3A
7A13 2102 0F
7A14 2103 09D2'DA
7A17 2104
7A17 2105 3C D3
7A19 2106
7A19 2107
7A19 2108
7A19 2109
7A19 2110
7A19 2111
7A19 2112
7A19 2113
7A19 2114
7A19 2115
7A19 2116
7A19 2117
7A19 2118
7A19 2119

*
*
*
*

*
*
*
*
*

6\$:

E 4

Fiche 1 Frame E4 Sequence 43
; RECEIVED DATA.

CALL TESTC_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 5\$;...JUMP TO BEGINNING OF ERROR LOOP.
LXI H,ONE_DAT ;LOAD ALL ONES INTO DATA0.
LXI D,DATA0
CALL MOVER_4
CALL FAST_WRITE_WRO ;WRITE ONES TO WRO.
LXI H,ZERO_DAT ;LOAD ALL ZEROS INTO DATA0.
LXI D,DATA0
CALL MOVER_4
CALL WRITE_WR1_32 ;WRITE ZEROS TO WR1.
CALL READ_WR1_32 ;READ WR1.
LXI H,ZERO_DAT ;MOVE ALL ZEROS INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_4
LXI H,DATA0 ;MOVE READ DATA INTO REC_DAT.
LXI D,REC_DAT
CALL MOVER_4
IFNEQ EXP_DAT,REC_DAT,4 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.
CALL TESTC_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 6\$;...JUMP TO BEGINNING OF ERROR LOOP.
TAIL

TESTD

32 BIT D AND Y BUSES TEST

THE REMAINING 24 BITS OF THE D AND Y BUSES THAT WERE NOT
TESTED BY THE 8 BIT LOADS WILL NOW BE TESTED. 32 BITS OF DATA WILL
BE LOADED INTO A 2901 WORKING REGISTER 8 BITS AT A TIME. THE DATA
WILL BE WRITTEN TO AN LS LOCATION AND THEN READ BACK. THE DATA USED
WILL BE THE STANDARD PATTERN OF A ONE RIPPLED THROUGH A FIELD OF
ZEROS AND THEN A ZERO RIPPLED THROUGH A FIELD OF ONES.

```

7A19 2120
7A19 2121
7A19 2122
7A19 2123
7A19 2124
7A19 2125
7A19 2126
7A19 2127
7A19 2128
7A19 2129
7A19 2130
7A19 2131
7A19 2132
7A19 2133
7A19 2134
7A19 2135
7A19 2136
7A19 2137
7A19 2138
7A19 2139
7A19 2140
7A19 2141
7A19 2142
7A19 2143
7A19 2144
7A19 2145
7A19 2146
7A19 2147
7A19 2148
7A19 2149
7A19 2150
7A19 2151
7A19 2152
7A19 2153
7A19 2154
7A19 2155
7A19 2156
7A19 2157 0000'CD 0D 3E
              OF 0000'3A
              C9 0A26'D2
              OF 0000'3A
              3D D3 0BAB'DA

```

```

7A2F 2158
7A2F 2159 00B9'CD
7A32 2160
7A32 2161
7A32 2162 00'54'21
7A35 2163 00'90'11
7A38 2164 0000'CD
7A3B 2165
7A3B 2166 00'54'21
7A3E 2167 00'00'11
7A41 2168 0000'CD
7A44 2169
7A44 2170 0172'CD
7A47 2171
7A47 2172 00'11'21
7A4A 2173 0294'CD
7A4D 2174
7A4D 2175 00'01'21

```

TEST STEPS:

- 1) LOAD WR WITH A DATA PATTERN.
- 2) WRITE THE PATTERN TO LS (MOV WR TO LS INST IN CSR).
- 3) CLEAR THE WORKING REGISTER.
- 4) READ THE DATA PATTERN BACK TO WR FROM LS (MOV LS TO WR).
- 5) READ WR BACK TO THE 8085 8 BITS AT A TIME.
- 6) PRINT ERROR IF ANY.
- 7) REPEAT 1-6 FOR EACH DATA PATTERN.

ASSUMPTIONS: THE CLEAR WR INSTRUCTION FROM THE PREVIOUS TEST IS WORKING AND THAT THE WR IS WORKING ALSO FROM PREVIOUS TESTING.

ERROR1: 32 BIT D AND Y BUS TEST FAILURE

- DEBUG:
- 1) CHECK THE Y BUS TO SEE THAT THE DATA REACHES LS FROM THE 2901.
 - 2) CHECK LS TO SEE THAT THE LOCATION USED(0) IS CORRECT ON 'LS ADRS L'.
 - 3) CHECK 'LS WRT EN L' AND 'ENABLE LONG H'.
 - 4) CHECK THE D BUS TO SEE THAT THE DATA RETURNING FROM LS IS CORRECT.
 - 5) CHECK 'ENABLE LS HI L' FOR LOW ON THE READ.
 - 6) REMOVE OTHER MODULES ON THE Y-BUS IF THE BUS APPEARS TO BE HELD DOWN.

TESTD: HEAD <^X0D>

1\$:

```

CALL LD_SUPPORT ;LOAD SUPPORT CODE FOR FAST READS AND
                ; WRITES OF WRO.

LXI H,ZERO DAT ;LOAD ALL ZEROS INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_4

LXI H,ZERO DAT ;LOAD ALL ZEROS DATA0.
LXI D,DATA0
CALL MOVER_4

CALL FAST_WRITE_WRO ;WRITE THE PATTERN TO WRO.

LXI H,MOV_WRO_LSO ;MOVE WRO TO LSO.
CALL EXEC_INST

LXI H,X_CLR_WRO+1 ;CLEAR WRO.

```

```

ZZ-ENKBB-2.0 7000 4
7A50 2176 0294'CD
7A53 2177
7A53 2178 00'14'21
7A56 2179 0294'CD
7A59 2180
7A59 2181 01B9'CD
7A5C 2182
7A5C 2183 00'00'21
7A5F 2184 00'95'11
7A62 2185 0C00'CD
7A65 2186
7A65 2187 00'90'21
          04 0E 00'95'11
          0A76'CA 0000'CD

7A73 2188
7A73 2189
7A73 2190 03E2'CD
7A76 2191
7A76 2192
7A76 2193
7A76 2194 0000'3A
7A79 2195 0F
7A7A 2196 0A3B'DA
7A7D 2197
7A7D 2198
7A7D 2199 00'59'21
7A80 2200 00'90'11
7A83 2201 0000'CD
7A86 2202
7A86 2203 00'59'21
7A89 2204 00'00'11
7A8C 2205 0000'CD
7A8F 2206
7A8F 2207 0172'CD
7A92 2208
7A92 2209 00'11'21
7A95 2210 0294'CD
7A98 2211
7A98 2212 00'01'21
7A9B 2213 0294'CD
7A9E 2214
7A9E 2215 00'14'21
7AA1 2216 0294'CD
7AA4 2217
7AA4 2218 01B9'CD
7AA7 2219
7AA7 2220 00'00'21
7AAA 2221 00'95'11
7AAD 2222 0000'CD
7AB0 2223
7AB0 2224 00'90'21
          04 0E 00'95'11
          0AC1'CA 0000'CD

7ABE 2225
7ABE 2226
7ABE 2227 03E2'CD
7AC1 2228
7AC1 2229
7AC1 2230
7AC1 2231 0000'3A

```

```

*****
*
*
*
*
*
*****

```

2\$:

```

*****
*
*
*
*
*
*****

```

```

G 4
Fiche 1 Frame G4 Sequence 45

CALL EXEC_INST

LXI H,MOV_LSO_WRO ;MOVE LSO TO WRO.
CALL EXEC_INST

CALL FAST_READ_WRO ;READ WRO.

LXI H,DATA0 ;MOVE THE RECEIVED DATA TO REC_DAT.
LXI D,REC_DAT
CALL MOVER_4

IFNEQ EXP_DAT,REC_DAT,4 ;COMPARE THE EXPECTED AND

; RECEIVED DATA.

CALL TESTD_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1$ ;...JUMP TO BEGINNING OF ERROR LOOP.

LXI H,ONE_DAT ;LOAD ALL ONES INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_4

LXI H,ONE_DAT ;LOAD THE PATTERN INTO DATA0.
LXI D,DATA0
CALL MOVER_4

CALL FAST_WRITE_WRO ;WRITE THE PATTERN TO WRO.

LXI H,MOV_WRO_LSO ;MOVE WRO TO LSO.
CALL EXEC_INST

LXI H,X_CLR_WRO+1 ;CLEAR WRO.
CALL EXEC_INST

LXI H,MOV_LSO_WRO ;MOVE LSO TO WRO.
CALL EXEC_INST

CALL FAST_READ_WRO ;READ WRO.

LXI H,DATA0 ;MOVE THE RECEIVED DATA TO REC_DAT.
LXI D,REC_DAT
CALL MOVER_4

IFNEQ EXP_DAT,REC_DAT,4 ;COMPARE THE EXPECTED AND

; RECEIVED DATA.

CALL TESTD_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...

```

```

ZZ-ENKBB-2.0 7000 4
7AC4 2232 0F
7AC5 2233 0A86'DA
7AC8 2234
7AC8 2235
7AC8 2236 00'55'21
7ACB 2237 00'8B'11
7ACE 2238 0000'CD
7AD1 2239
7AD1 2240 C5 46 00'00'21
7AD1 2240 77 20 3E
7AD9 2241
7AD9 2242 00'8B'21
7ADC 2243 00'90'11
7ADF 2244 0000'CD
7AE2 2245
7AE2 2246 00'8B'21
7AE5 2247 00'00'11
7AE8 2248 0000'CD
7AEB 2249
7AEB 2250 0172'CD
7AEE 2251
7AEE 2252 00'11'21
7AF1 2253 0294'CD
7AF4 2254
7AF4 2255 00'01'21
7AF7 2256 0294'CD
7AFA 2257
7AFA 2258 00'14'21
7AFD 2259 0294'CD
7B00 2260
7B00 2261 01B9'CD
7B03 2262
7B03 2263 00'00'21
7B06 2264 00'95'11
7B09 2265 0000'CD
7B0C 2266
7B0C 2267 00'90'21
7B0C 2267 04 0E 00'95'11
7B0C 2267 0B1D'CA 0000'CD
7B1A 2268
7B1A 2269
7B1A 2270 03E2'CD
7B1D 2271
7B1D 2272
7B1D 2273
7B1D 2274 0000'3A
7B20 2275 0F
7B21 2276 0AE2'DA
7B24 2277
7B24 2278 B7
7B25 2279
7B25 2280 00'8A'21
7B28 2281 0000'22
7B2B 2282 0000'CD
7B2E 2283
7B2E 2284 35 00'00'21
7B2E 2284 70 C1 0AD9'C2
7B37 2285
7B37 2286 00A9'CD
7B3A 2287

```

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

3\$:

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

H 4

Fiche 1 Frame H4

Sequence 46

RRC
JC

2\$

;...JUMP TO BEGINNING OF ERROR LOOP.

LXI H,ONE_SHIFT
LXI D,TEMPB_DAT
CALL MOVER_4

;LOAD A ONE THROUGH A ZERO FIELD
; PATTERN INTO TEMPB_DAT.

DO 32

LXI H,TEMPB_DAT
LXI D,EXP_DAT
CALL MOVER_4

;LOAD THE PATTERN INTO EXP_DAT.

LXI H,TEMPB_DAT
LXI D,DATA0
CALL MOVER_4

;LOAD THE PATTERN INTO DATA0.

CALL FAST_WRITE_WRO

;WRITE THE PATTERN TO WRO.

LXI H,MOV_WRO_LSO
CALL EXEC_INST

;MOVE WRO TO LSO.

LXI H,X_CLR_WRO+1
CALL EXEC_INST

;CLEAR WRO.

LXI H,MOV_LSO_WRO
CALL EXEC_INST

;MOVE LSO TO WRO.

CALL FAST_READ_WRO

;READ WRO.

LXI H,DATA0
LXI D,REC_DAT
CALL MOVER_4

;MOVE THE RECEIVED DATA TO REC_DAT.

IFNEQ EXP_DAT,REC_DAT,4 ;COMPARE THE EXPECTED AND

; RECEIVED DATA.

CALL TESTD_ERROR

;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON
RRC
JC 3\$

;IF THE ERROR LOOPING FLAG IS SET...

;...JUMP TO BEGINNING OF ERROR LOOP.

ORA A

;CLEAR THE CARRY.

LXI H,TEMPB_SHIFT
SHLD SHIFT_PNT
CALL SHIFTER

;SHIFT THE ONE THROUGH THE ZERO FIELD.

ENDDO

CALL CHECK_RD

;CALL BOOT CODE IF UNDER RD CONTROL

ZZ-ENKBB-2.0 7000 4

```
7B3A 2288
7B3A 2289 00'5A'21
7B3D 2290 00'8B'11
7B40 2291 0000'CD
7B43 2292
7B43 2293 C5 46 00'00'21 *****
          77 20 3E
7B4B 2294
7B4B 2295 00'8B'21
7B4E 2296 0C'90'11
7B51 2297 0000'CD
7B54 2298
7B54 2299 00'8B'21
7B57 2300 00'00'11
7B5A 2301 0000'CD
7B5D 2302
7B5D 2303 01'72'CD
7B60 2304
7B60 2305 00'11'21
7B63 2306 02'94'CD
7B66 2307
7B66 2308 00'01'21
7B69 2309 02'94'CD
7B6C 2310
7B6C 2311 00'14'21
7B6F 2312 02'94'CD
7B72 2313
7B72 2314 01'89'CD
7B75 2315
7B75 2316 00'00'21
7B78 2317 00'95'11
7B7B 2318 0000'CD
7B7E 2319
7B7E 2320 00'90'21 *****
          04 0E 00'95'11
          0B8F'CA 0000'CD
7B8C 2321
7B8C 2322
7B8C 2323 03E2'CD
7B8F 2324
7B8F 2325 *****
7B8F 2326
7B8F 2327 0000'3A
7B92 2328 0F
7B93 2329 0B54'DA
7B96 2330
7B96 2331 37
7B97 2332
7B97 2333 00'8A'21
7B9A 2334 0000'22
7B9D 2335 0000'CD
7BA0 2336
7BA0 2337 35 00'00'21 *****
          70 C1 0B4B'C2
7BA9 2338
7BA9 2339 3C D3
7BAB 2340
7BAB 2341
7BAB 2342
7BAB 2343
```

4\$:

I 4

Fiche 1 Frame I4

Sequence 47

```
LXI H,ZERO SHIFT ;LOAD A ZERO THROUGH A ONE FIELD
LXI D,TEMPB_DAT ; PATTERN INTO TEMPB_DAT.
CALL MOVER_4

DO 32

LXI H,TEMPB_DAT ;LOAD THE PATTERN INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_4

LXI H,TEMPB_DAT ;LOAD THE PATTERN INTO DATA0.
LXI D,DATA0
CALL MOVER_4

CALL FAST_WRITE_WRO ;WRITE THE PATTERN TO WRO.

LXI H,MOV_WRO_LSO ;MOVE WRO TO LSO.
CALL EXEC_INST

LXI H,X CLR_WRO+1 ;CLEAR WRO.
CALL EXEC_INST

LXI H,MOV_LSO_WRO ;MOVE LSO TO WRO.
CALL EXEC_INST

CALL FAST_READ_WRO ;READ WRO.

LXI H,DATA0 ;MOVE THE RECEIVED DATA TO REC_DAT.
LXI D,REC_DAT
CALL MOVER_4

IFNEQ EXP_DAT,REC_DAT,4 ;COMPARE THE EXPECTED AND
                          ; RECEIVED DATA.

CALL TESTD_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 4$ ;...JUMP TO BEGINNING OF ERROR LOOP.

STC ;SET THE CARRY.

LXI H,TEMPB_SHIFT ;SHIFT THE ZERO THROUGH THE ONE FIELD.
SHLD SHIFT_PNT
CALL SHIFTER

ENDDO

TAIL
```

;*****

7BAB 2344
7BAB 2345
7BAB 2346
7BAB 2347
7BAB 2348
7BAB 2349
7BAB 2350
7BAB 2351
7BAB 2352
7BAB 2353
7BAB 2354
7BAB 2355
7BAB 2356
7BAB 2357
7BAB 2358
7BAB 2359
7BAB 2360
7BAB 2361
7BAB 2362
7BAB 2363
7BAB 2364
7BAB 2365
7BAB 2366
7BAB 2367
7BAB 2368
7BAB 2369
7BAB 2370
7BAB 2371
7BAB 2372
7BAB 2373
7BAB 2374
7BAB 2375
7BAB 2376
7BAB 2377
7BAB 2378
7BAB 2379
7BAB 2380
7BAB 2381
7BAB 2382
7BAB 2383
7BAB 2384
7BAB 2385
7BAB 2386
7BAB 2387
7BAB 2388
7BAB 2389
7BAB 2390
7BAB 2391
7BAB 2392
7BAB 2393
7BAB 2394
7BAB 2395
7BAB 2396
7BAB 2397
7BAB 2398
7BAB 2399
7BAB 2400
7BAB 2401
7BAB 2402
7BAB 2403

0000'CD 0E 3E

TESTE

LS RAM DATA TEST

THE LS RAM DATA TEST TESTS EACH LOCATION OF THE RAM SO THAT EACH 4 BIT SLICE GETS TESTED WITH RIPPLED 0 THROUGH 1'S AND ONE RIPPLED THROUGH ZEROS. THE LOCATIONS TO BE TESTED ARE FROM 0 TO F7 EXCLUDING 78 - 7F. THIS IS DUE TO THE FACT THE UNTESTED LOCATIONS HAVE SPECIAL FUNCTIONS AND DO NOT ACTUALLY USE THE LS RAM (F8 - FF ARE ALSO IN THE CATEGORY).

DATA PATTERNS: FFFFFFFF
EEEEEEEE
DDDDDDDD
BBBBBBBB
77777777
00000000
11111111
22222222
44444444
88888888

TEST STEPS:

- 1) WRITE DATA PATTERN TO WR.
- 2) MOVE DATA FROM WR TO LS.
- 3) CLEAR WR.
- 4) READ DATA FROM LS TO WR.
- 5) COMPARE DATA SENT WITH RETURNED.
- 6) PRINT ERROR IF ANY.
- 7) REPEAT 1 - 6 FOR ALL DATA PATTERNS.
- 9) REPEAT 1 - 7 FOR ALL LS LOCATIONS.

ASSUMPTIONS: THE CLEAR WR INSTRUCTION FROM THE PREVIOUS TEST IS WORKING, THAT THE WR IS WORKING ALSO FROM PREVIOUS TESTING, THAT THE DATA PATH TO LS IS WORKING FROM THE PREVIOUS TEST.

ERROR1: LS RAM DATA TEST FAILURE.

NOTE: ADDITIONAL DATA CONTAINS THE PRESENT LS LOCATION.

DEBUG: 1) CHECK RAM MEMORY FOR ERRORS.
2) ALTHOUGH EACH ADDRESS IN THE LS IS ACCESSED IN MOST CASES IF THERE WHERE AN ADDRESS FAILURE IT WOULD GO UNDETECTED BY THIS TEST. HOWEVER IF AN ADDRESS SHOULD END UP IN THE RANGE THAT CAUSES SPECIAL FUNCTIONS SUCH AN ERROR COULD OCCUR, SO 'LS ADRS 0 - 7 H' SHOULD BE CHECKED.

TESTE:

HEAD

<^XOE>

Sequence 49

```

CALL      LD_SUPPORT      ;LOAD SUPPORT FOR FAST READS AND
                        ; WRITES OF WRO.

MVI      A,1              ;FIRST (AND ONLY) PART OF TEST E.
STA      CON_ERR_NUM

MVI      A,0              ;LS_ADDR POINTS TO THE LS LOACTION
STA      LS_ADDR          ; BEING WRITTEN AND READ.

MVI      A,20             ;INITIALIZE LOOP_CNT.
STA      LOOP_CNT

LXI      H,TESTE_DAT      ;POINT AT THE FIRST OF 10 TEST DATA
SHLD     DATA_PNT        ; PATTERNS.

DO        10              ;TEN PATTERNS ARE WRITTEN TO LS.

LHLD     DATA_PNT        ;MOVE A TEST PATTERN TO EXP_DAT.
LXI      D,EXP_DAT
CALL     MOVER_4

LHLD     DATA_PNT        ;MOVE A PATTERN TO DATA0.
LXI      D,DATA0
CALL     MOVER_4

LDA      LS_ADDR          ;MOVE LS POINTER TO ACCUMULATOR.
CALL     LS_WRITE         ;WRITE TO LS.

LDA      LS_ADDR          ;MOVE LS POINTER TO ACCUMULATOR.
CALL     LS_READ          ;READ FROM LS.

IFNEQ    EXP_DAT,DATA0,4   ;COMPARE THE EXPECTED AND
                        ; RECEIVED DATA.

CALL     TESTE_F_ERROR    ;PRINT AN ERROR MESSAGE.

ENDIF

LDA      ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC       2$              ;...JUMP TO BEGINNING OF ERROR LOOP.

LHLD     DATA_PNT        ;POINT AT THE NEXT DATA PATTERN.
INX      H
INX      H
INX      H
INX      H
SHLD     DATA_PNT

ENDDO

```

ZZ-ENKBB-2.0 7000 4

L 4

Fiche 1 Frame L4

Sequence 50

7C2A 2456
7C2A 2457 00'0C'21
7C2D 2458 35
7C2E 2459 0C40'C2
7C31 2460
7C31 2461 14 3E
7C33 2462 000C'32
7C36 2463
7C36 2464 0000'CD
7C39 2465 0C00'3A
7C3C 2466 0F
7C3D 2467 0099'DC
7C40 2468
7C40 2469 0009'3A
7C43 2470 F7 FE
7C45 2471 0C5C'CA
7C48 2472
7C48 2473 78 FE
7C4A 2474 0C54'CA
7C4D 2475
7C4D 2476 3C
7C4E 2477 0009'32
7C51 2478
7C51 2479 0BD3'C3
7C54 2480
7C54 2481 80 3E
7C56 2482 0009'32
7C59 2483
7C59 2484 0BD3'C3
7C5C 2485
7C5C 2486 3C D3
7C5E 2487
7C5E 2488
7C5E 2489
7C5E 2490
7C5E 2491
7C5E 2492
7C5E 2493
7C5E 2494
7C5E 2495
7C5E 2496
7C5E 2497
7C5E 2498
7C5E 2499
7C5E 2500
7C5E 2501
7C5E 2502
7C5E 2503
7C5E 2504
7C5E 2505
7C5E 2506
7C5E 2507
7C5E 2508
7C5E 2509
7C5E 2510
7C5E 2511
7C5E 2512
7C5E 2513
7C5E 2514
7C5E 2515

3\$:

4\$:

5\$:

```
LXI    H,LOOP_CNT    ;POINT TO LOOP CNT.
DCR    M              ;DECREMENT LOOP CNT.
JNZ    3$             ;IF NOT ZERO, DON'T CHECK FOR ^C.

MVI    A,20           ;INITIALIZE LOOP_CNT.
STA    LOOP_CNT

CALL   GCVECT         ;IF CONTROL C HAS BEEN TYPED...
LDA    CNTLC_TYPED
RRC
CC      CON_SET_FOR_CO ;...GO TO MONITOR.

LDA    LS_ADDR        ;HAS ALL LS BEEN TESTED.
CPI    <^XF7>
JZ     5$

CPI    <^X78>         ;SKIP LS 70-7F AS THEY ARE USED
JZ     4$             ; FOR SPECIAL PURPOSES.

INR    A              ;INCPMENT THE LS_POINTER.
STA    LS_ADDR

JMP    1$

MVI    A,<^X80>        ;MOVE HEX 80 TO THE LS POINTER AND
STA    LS_ADDR         ; RESUME TESTING.

JMP    1$

TAIL
```

TESTF

LS RAM MARCH PATTERN TEST

THE RAM MARCH PATTERN TEST WRITES A BACKGROUND OF ZEROS THROUGH THE MEMORY THEN GOES TO EACH LOCATION WORKING ITS WAY UP READING THE 0 BACKGROUND, WRITING A 1 AND READING THE ONE BACK. THIS THEN LEAVES 1'S IN THE BACKGROUND AND THE TEST WILL START AT THE HIGH ADDRESS AND READ 1'S, WRITE 0'S, READ 0'S. THE MEMORY WILL BE TESTED FROM 0 TO F7 EXCLUDING 78 - 7F. THIS IS DUE TO THE FACT THE UNTESTED LOCATIONS HAVE SPECIAL FUNCTIONS AND DO NOT ACTUALLY USE THE LS RAM (F8 - FF ARE ALSO IN THE CATEGORY)

TEST STEPS:

- 1) WRITE BACKGROUND OF ZEROS.
- 2) READ AND COMPARE TO ZERO.
- 3) PRINT ERROR IF ANY.
- 4) WRITE 1, READ, AND COMPARE TO 1.
- 5) PRINT ERROR IF ANY.
- 6) REPEAT 2-5 FOR ALL LOCATIONS (LOW TO HIGH).
- 7) READ AND COMPARE TO ONE.
- 8) PRINT ERROR IF ANY.

ZZ-ENKBB-2.0 7000 4

7C5E 2516
7C5E 2517
7C5E 2518
7C5E 2519
7C5E 2520
7C5E 2521
7C5E 2522
7C5E 2523
7C5E 2524
7C5E 2525
7C5E 2526
7C5E 2527
7C5E 2528
7C5E 2529
7C5E 2530
7C5E 2531
7C5E 2532
7C5E 2533
7C5E 2534
7C5E 2535
7C5E 2536
7C5E 2537
7C5E 2538
7C5E 2539

0000'CD OF 3E
OF 0000'3A
C9 0C6B'D2
OF 0000'3A
3D D3 0DC6'DA

7C74 2540
7C74 2541 00B9'CD
7C77 2542
7C77 2543
7C77 2544
7C77 2545 00 3E
7C79 2546 0009'32
7C7C 2547
7C7C 2548 00'54'21
7C7F 2549 00'00'11
7C82 2550 0000'CD
7C85 2551
7C85 2552 0009'3A
7C88 2553 0138'CD
7C8B 2554
7C8B 2555 0009'3A
7C8E 2556 F7 FE
7C90 2557 0CA7'CA
7C93 2558
7C93 2559 77 FE
7C95 2560 0C9F'CA
7C98 2561
7C98 2562 3C
7C99 2563 0009'32
7C9C 2564 0C7C'C3
7C9F 2565
7C9F 2566 80 3E
7CA1 2567 0009'32
7CA4 2568 0C7C'C3
7CA7 2569
7CA7 2570
7CA7 2571

M 4

Fiche 1 Frame M4

Sequence 51

- 9) WRITE 0, READ, AND COMPARE TO 0.
10) PRINT ERROR IF ANY.
11) REPEAT 7-10 FOR ALL LOCATIONS (HIGH TO LOW).

ASSUMPTIONS: THE DATA PATH TO AND FROM LS IS WORKING
FROM PREVIOUS TESTS.

ERROR1: LS RAM MARCH TEST FAILURE.

NOTE: ADDITIONAL DATA CONTAINS PRESENT LS LOACTION.

DEBUG: 1) CHECK ADDRESS DECODE FOR LS FROM THE MUX OF CSR BITS
AND OS BITS. THE CSR BITS SHOULD BE ENABLED THROUGH.
2) CHECK 'LS ADRS 0 - 7 H' FOR ADDRESSING ERRORS.
3) CHECK RAM MEMORY FOR ERRORS IN INTERNAL ADDRESS LOGIC.

TESTF:	HEAD	<^X0F>	
	CALL	LD_SUPPORT	;LOAD SUPPORT FOR FAST READS AND ; WRITES OF WRO.
	MVI	A,0	;ZERO LS POINTER.
	STA	LS_ADDR	
1\$:	LXI	H,ZERO DAT	;LOAD ALL ZEROS INTO DATA0.
	LXI	D,DATA0	
	CALL	MOVER_4	
	LDA	LS_ADDR	;MOV LS POINTER TO ACCUM.
	CALL	LS_WRITE	;WRITE ALL ZEROS TO LS.
	LDA	LS_ADDR	;HAS ALL LS BEEN WRITTEN?
	CPI	<^XF7>	
	JZ	3\$	
	CPI	<^X77>	;HAS LS_POINTER REACHED BLOCK ; OF LS TO BE SKIPPED.
	JZ	2\$	
	INR	A	;INCREMENT LS POINTER.
	STA	LS_ADDR	
	JMP	1\$	;WRITE TO NEXT LS LOCATION.
2\$:	MVI	A,<^X80>	;MOVE HEX 80 TO LS POINTER (SKIP ; LOCATIONS 78-7F).
	STA	LS_ADDR	
	JMP	1\$	;WRITE TO NEXT LS LOCATION.

```

ZZ-ENKBB-2.0 7000 4
7CA7 2572 0000'CD
7CAA 2573 0000'3A
7CAD 2574 0F
7CAE 2575 0099'DC
7CB1 2576
7CB1 2577
7CB1 2578
7CB1 2579 01 3E
7CB3 2580 0000'32
7CB6 2581
7CB6 2582 00 3E
7CB8 2583 0009'32
7CBB 2584
7CBB 2585 0009'3A
7CBE 2586 0155'CD
7CC1 2587
7CC1 2588 00'54'21
04 0E 00'00'11
0CDB'CA 0000'CD
7CCF 2589
7CCF 2590
7CCF 2591 00'54'21
7CD2 2592 00'90'11
7CD5 2593 0000'CD
7CD8 2594
7CD8 2595 040D'CD
7CDB 2596
7CDB 2597
7CDB 2598
7CDB 2599 0000'3A
7CDE 2600 0F
7CDF 2601 0CBB'DA
7CE2 2602
7CE2 2603 00'59'21
7CE5 2604 00'00'11
7CE8 2605 0000'CD
7CEB 2606
7CEB 2607 0009'3A
7CEE 2608 0138'CD
7CF1 2609
7CF1 2610 04 0E 00'00'21
0D 23 77 AF
0CF7'C2
7CFD 2611
7CFD 2612 0009'3A
7D00 2613 0155'CD
7D03 2614
7D03 2615 00'59'21
04 0E 00'00'11
0D1D'CA 0000'CD
7D11 2616
7D11 2617
7D11 2618 00'59'21
7D14 2619 00'90'11
7D17 2620 0000'CD
7D1A 2621
7D1A 2622 040D'CD
7D1D 2623
7D1D 2624
7D1D 2625

```

3\$:

4\$:

5\$:

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

N 4

Fiche 1 Frame N4 Sequence 52

```

CALL GCVECT ;IF CONTROL C HAS BEEN TYPED...
LDA CNTLC_TYPED
RRC
CC CON_SET_FOR_CO ;...GO TO MONITOR.

MVI A,1 ;FIRST PART OF TEST F.
STA CON_ERR_NUM

MVI A,0 ;ZERO LS POINTER.
STA LS_ADDR

LDA LS_ADDR ;MOVE LS POINTER TO A.
CALL LS_READ ;READ LS.

IFNEQ ZERO_DAT,DATA0,4 ;COMPARE DATA READ WITH

; ALL ZEROS.

LXI H,ZERO_DAT ;LOAD ZEROS INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_4

CALL TESTE_F_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 4$ ;...JUMP TO BEGINNING OF ERROR LOOP.

LXI H,ONE_DAT ;LOAD ALL ONES INTO DATA0.
LXI D,DATA0
CALL MOVER_4

LDA LS_ADDR ;LOAD LS POINTER INTO ACCUM.
CALL LS_WRITE ;WRITE TO LS.

ZERO DATA0,4 ;CLEAR DATA0.

LDA LS_ADDR ;LOAD LS POINTER INTO ACCUM.
CALL LS_READ ;READ LS.

IFNEQ ONE_DAT,DATA0,4 ;COMPARE THE EXPECTED AND

; RECEIVED DATA.

LXI H,ONE_DAT ;LOAD ALL ONES INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_4

CALL TESTE_F_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF

```

```

ZZ-ENKBB-2.0      7000      4
7D1D 2626 0000'3A
7D20 2627 0F
7D21 2628 0CE2'DA
7D24 2629
7D24 2630
7D24 2631 0009'3A
7D27 2632 F7 FE
7D29 2633 0D40'CA
7D2C 2634
7D2C 2635 77 FE
7D2E 2636 0D38'CA
7D31 2637
7D31 2638 3C
7D32 2639 0009'32
7D35 2640 0CBB'C3
7D38 2641
7D38 2642 80 3E      6$:
7D3A 2643 0009'32
7D3D 2644 0CBB'C3
7D40 2645
7D40 2646
7D40 2647
7D40 2648 0000'CD      7$:
7D43 2649 0000'3A
7D46 2650 0F
7D47 2651 0099'DC
7D4A 2652
7D4A 2653
7D4A 2654 02 3E
7D4C 2655 0000'32
7D4F 2656
7D4F 2657 F7 3E
7D51 2658 0009'32
7D54 2659
7D54 2660 0009'3A      8$:
7D57 2661 0155'CD
7D5A 2662
7D5A 2663 00'59'21
              04 0E 00'00'11
              0D74'CA 0000'CD
              *****
7D68 2664
7D68 2665
7D68 2666 00'59'21
7D6B 2667 00'90'11
7D6E 2668 0000'CD
7D71 2669
7D71 2670 040D'CD
7D74 2671
7D74 2672
              *****
7D74 2673
7D74 2674 0000'3A
7D77 2675 0F
7D78 2676 0D54'DA
7D7B 2677
7D7B 2678 00'54'21
7D7E 2679 00'00'11
7D81 2680 0000'CD
7D84 2681
7D84 2682 0009'3A
7D87 2683 0138'CD

```

B 5

```

LDA  ERROR_CON      Fiche 1 Frame B5      Sequence 53
RRC                                     ;IF THE ERROR LOOPING FLAG IS SET...
JC   5$                                     ;...JUMP TO BEGINNING OF ERROR LOOP.

LDA  LS_ADDR
CPI  <^XF7>          ;HAVE ALL LS LOCATIONS BEEN TESTED
JZ   7$               ; FOR FIRST MAIN PASS?

CPI  <^X77>          ;HAS LS POINTER REACHED LS BLOCK
JZ   6$               ; TO BE SKIPPED?

INR  A
STA  LS_ADDR          ;INCREMENT THE LS POINTER.
JMP  4$               ;TEST NEXT LS LOCATION.

MVI  A,<^X80>         ;MOVE HEX 80 TO LS POINTER (SKIP
STA  LS_ADDR          ; LOCATIONS 78-7F).
JMP  4$               ;TEST NEXT LS LOCATION.

CALL GCVECT           ;IF CONTROL C HAS BEEN TYPED...
LDA  CNTLC_TYPED
RRC
CC   CON_SET_FOR_CO   ;...GO TO MONITOR.

MVI  A,2
STA  CON_ERR_NUM      ;SECOND PART OF TEST F.

MVI  A,<^XF7>         ;POINT TO TOP OF LS.
STA  LS_ADDR

LDA  LS_ADDR          ;MOVE LS POINTER TO A.
CALL LS_READ          ;READ LS.

IFNEQ ONE_DAT,DATA0,4 ;COMPARE DATA READ WITH

                                ; ALL ONES.

LXI  H,ONE_DAT        ;LOAD ALL ONES INTO EXP_DAT.
LXI  D,EXP_DAT
CALL MOVER_4

CALL TESTE_F_ERROR    ;PRINT AN ERROR MESSAGE.

ENDIF

LDA  ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...
RRC                                     ;...JUMP TO BEGINNING OF ERROR LOOP.
JC   8$

LXI  H,ZERO_DAT      ;LOAD ALL ZEROS INTO DATA0.
LXI  D,DATA0
CALL MOVER_4

LDA  LS_ADDR          ;MOVE LS POINTER TO A.
CALL LS_WRITE         ;WRITE TO LS.

```

ZZ-ENKBB-2.0 7000 4

7D8A 2684
 7D8A 2685 0009'3A
 7D8D 2686 0155'CD
 7D90 2687
 7D90 2688 00'54'21
 04 0E 00'00'11
 ODA1'CA 0000'CD
 7D9E 2689
 7D9E 2690
 7D9E 2691 040D'CD
 7DA1 2692
 7DA1 2693
 7DA1 2694
 7DA1 2695 0000'3A
 7DA4 2696 0F
 7DA5 2697 0D7B'DA
 7DA8 2698
 7DA8 2699 0009'3A
 7DAB 2700 00 FE
 7DAD 2701 0DC4'CA
 7DB0 2702
 7DB0 2703 80 FE
 7DB2 2704 0DBC'CA
 7DB5 2705
 7DB5 2706 3D
 7DB6 2707 0009'32
 7DB9 2708 0D54'C3
 7DBC 2709
 7DBC 2710 77 3E
 7DBE 2711 0009'32
 7DC1 2712 0D54'C3
 7DC4 2713
 7DC4 2714 3C D3
 7DC6 2715
 7DC6 2716
 7DC6 2717
 7DC6 2718
 7DC6 2719
 7DC6 2720
 7DC6 2721
 7DC6 2722
 7DC6 2723
 7DC6 2724 0000'CD 51 3E
 0F 0000'3A
 C9 0DD3'D2
 0F 0000'3A
 3D D3 ODDE'DA
 7DDC 2725
 7DDC 2726 3C D3
 7DDE 2727

 *
 *
 *
 *
 *

C 5

Fiche 1 Frame C5

Sequence 54

LDA LS_ADDR ;MOVE LS POINTER TO A.
 CALL LS_READ ;READ LS.
 IFNEQ ZERO_DAT,DATA0,4 ;COMPARE THE EXPECTED AND
 ; RECEIVED DATA.
 CALL TESTE_F_ERROR ;PRINT AN ERROR MESSAGE.
 ENDIF
 LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
 RRC
 JC 9\$;...JUMP TO BEGINNING OF ERROR LOOP.
 LDA LS_ADDR ;HAVE ALL LS LOCATIONS BEEN TESTED?
 CPI <^X00>
 JZ 11\$
 CPI <^X80> ;HAS LS POINTER REACHED LS BLOCK
 JZ 10\$; TO BE SKIPPED?
 DCR A ;DECREMENT THE LS POINTER.
 STA LS_ADDR
 JMP 8\$;TEST NEXT LS LOCATION.
 10\$: MVI A,<^X77> ;MOVE HEX 80 TO LS POINTER (SKIP
 STA LS_ADDR ; LOCATIONS 78-7F).
 JMP 8\$;TEST NEXT LS LOCATION.

10\$:

11\$:

TAIL

 :
 : LAST TEST FOR EVERY CONSOL_SECTION....FORCES END OF SECTION
 :
 :*****

END_TEST: HEAD <^X51>

TAIL

\$PSW	=	00000006		
\$SP	=	00000006		
A	=	00000007		
ALLOWP	=	00000003		
APTFLG	=	00000004		
AUTO	=	00000005		
B	=	00000000		
BOOTEN	=	00000007		
BOOTF	=	00000001		
C	=	00000001		
CHECK_RD		000000A9	R	02
CLRACK	=	000000A9		
CLRACL	=	00000033		
CLRATN	=	000000AB		
CLRBSY	=	0000002E		
CLRCLK	=	00000020		
CLRCSK	=	00000024		
CLRCSR	=	00000038		
CLRDCL	=	00000031		
CLRHLT	=	000000AF		
CLRLIT	=	0000003C		
CLRMCI	=	00000029		
CLRMCK	=	0000002B		
CLRPFI	=	000000AD		
CLRSS	=	00000022		
CLRTIM	=	000000A5		
CLRTMR	=	0000002C		
CLRUPC	=	000000A8		
CLRUPK	=	00000026		
CLRWCK	=	00000036		
CNT	=	0000081A		
CNTA	=	00000FA0		
CNTLC_TYPED		*****	X	02
COMPARE		*****	X	02
CON_ADD_DAT		*****	X	02
CON_BEGIN_TEST		*****	X	02
CON_ERROR		*****	X	02
CON_ERR_NUM		*****	X	02
CON_EXP_DAT		*****	X	02
CON_MOD_DAT		*****	X	02
CON_MSK_DAT		*****	X	02
CON_REC_DAT		*****	X	02
CON_SET_FOR_CO		00000099	R	02
CPATTN	=	00000083		
CPUACK	=	00000082		
CSR07	=	00000085		
CSR15	=	00000086		
CSR23	=	00000087		
CSR_SHIFT		*****	X	02
CSR_VALUE		*****	X	02
D	=	00000002		
DATA0		*****	X	02
DATA1		*****	X	02
DATA2		*****	X	02
DATA3		*****	X	02
DATA_PNT		0000000A	R	02
DATA_SHIFT		*****	X	02

DCLO	=	00000002		
DISMEM	=	000000A7		
DISPAR	=	000000A1		
DOLOOP		*****	X	02
E	=	00000003		
ENBMEM	=	000000A6		
ENBPAR	=	000000A0		
END_TEST		00000DC6	R	02
ENTRY		00000000	R	02
EOS_FLG		*****	X	02
ERROR_CON		*****	X	02
EXEC_INST		00000294	R	02
EXP_DAT		00000090	R	02
EXP_SHIFT		0000008F	R	02
FAST_READ_WRO		000001B9	R	02
FAST_WRITE_WRO		00000172	R	02
FILE_NAME		00000005	R	02
GCVECT		*****	X	02
H	=	00000004		
HEAD	=	00000000		
HLTFLG	=	00000002		
INC_WORD		*****	X	02
INITH	=	00000035		
INITL	=	00000034		
ITCSR	=	0000001D		
ITDB	=	0000001C		
L	=	00000005		
LD_CSR_INST		0000029C	R	02
LD_SUPPORT		000000B9	R	02
LITERAL	=	00000001		
LOAD_MOD_DAT		000002D2	R	02
LOAD_UPC		*****	X	02
LOOP_CNT		0000000C	R	02
LS_ADDR		00000009	R	02
LS_READ		00000155	R	02
LS_WRITE		00000138	R	02
LVL	=	00000000		
M	=	00000006		
MACSTK1	=	0000081A		
MACSTK2	=	00000818		
MACSTK3	=	00000811		
MCPU_A		*****	X	02
MICRO_STEP_CPU		*****	X	02
MIPFLG	=	000040D9		
MISC2	=	00000001		
MISC2_WR_CRR		0000002F	R	02
MOVER		*****	X	02
MOVER_3		*****	X	02
MOVER_4		*****	X	02
MOV_CCR_WRO		0000000E	R	02
MOV_LSO_WRO		00000014	R	02
MOV_LSX_WRO		0000001A	R	02
MOV_WRO_LSO		00000011	R	02
MOV_WRO_LSX		00000017	R	02
MOV_WR1_WR1		0000001D	R	02
MWCS_A		*****	X	02
NA_OTHER		*****	X	02

NOP_CSR	*****	X	02
OKVTS	= 00000007		
ONE_DAT	00000059	R	02
ONE_SHIFT	00000055	R	02
PARITY_VECTOR	= 00004C22		
POWER_VECTOR	= 00004C25		
READ	= 00000080		
READJ1	= 00000003		
READJ2	= 00000004		
READ_BYTE_WRO	000001D9	R	02
READ_WR1_32	0000023E	R	02
REC_DAT	00000095	R	02
REC_SHIFT	00000094	R	02
REMDIS	= 00000006		
REPORT_CPU_ERROR	00000446	R	02
REPORT_ERROR	0000044C	R	02
ROT_WRT_L	00000023	R	02
SAVE_LS_ADDR	0000000D	R	02
SETACK	= 000000A8		
SETACL	= 00000032		
SETATN	= 000000AA		
SETBSY	= 0000002F		
SETCLK	= 00000021		
SETCSK	= 00000025		
SETCSR	= 00000039		
SETDCL	= 00000030		
SETHLT	= 000000AE		
SETLIT	= 0000003D		
SETMCI	= 00000028		
SETMCK	= 0000002A		
SETPFI	= 000000AC		
SETSS	= 00000023		
SETTIM	= 000000A4		
SETTMR	= 0000002D		
SETUPC	= 000000A9		
SETUPK	= 00000027		
SETWCK	= 00000037		
SET_COMPARE	0000011B	R	02
SET_FOR_CONT	*****	X	02
SHIFTER	*****	X	02
SHIFTER1	*****	X	02
SHIFT_CNT	00000007	R	02
SHIFT_CNT_DAT	00000008	R	02
SHIFT_L_2901	*****	X	02
SHIFT_PNT	*****	X	02
SHIFT_WR1_L	00000026	R	02
SHIFT_WR1_R	00000020	R	02
SHIFT_WR1_R8	0000027C	R	02
SKIP_TEST	*****	X	02
SUPPORT_CODE	00000032	R	02
SYM	= 0000081A		
TEMPA_DAT	00000087	R	02
TEMPA_SHIFT	00000086	R	02
TEMPB_DAT	0000008B	R	02
TEMPB_SHIFT	0000008A	R	02
TEST9	00000450	R	02
TEST9_1_0	000002DB	R	02

TEST9_2_0	0000033F	R	02
TEST9_ERR2_3	00000380	R	02
TEST9_ERROR	0000037B	R	02
TEST9_SHIFT	00000307	R	02
TESTA	00000515	R	02
TESTA_ERROR	000003A0	R	02
TESTB	00000609	R	02
TESTB_ERROR	000003CD	R	02
TESTC	0000083E	R	02
TESTC_ERROR	000003E7	R	02
TESTD	00000A19	R	02
TESTD_ERROR	000003E2	R	02
TESTE	000008AB	R	02
TESTE_DAT	0000005E	R	02
TESTE_F_ERROR	0000040D	R	02
TESTF	00000C5E	R	02
TTCR	= 0000004B		
TTDB	= 00000048		
TTMODE	= 0000004A		
TTSR	= 00000049		
TUCR	= 00000047		
TUDB	= 00000044		
TUMODE	= 00000046		
TUSR	= 00000045		
UPC14	= 00000084		
UPC14A	= 00000000		
UPC_VALUE	*****	X	02
VERSION	00000003	R	02
VER_WRITE_TAB	00000044	R	02
VNORML	= 000000A3		
VSHIFT	= 000000A2		
WCSB0	= 00000008		
WCSB1	= 00000009		
WCSB2	= 0000000A		
WCS_ADDRESS	*****	X	02
WCS_VALUE	*****	X	02
WCS_WRITE	*****	X	02
WRITE	= 000000CC		
WRITE_TAB_PNT	*****	X	02
WRITE_WR1_32	00000205	R	02
XD_ADDR	*****	X	02
X_CLR_WRO	*****	X	02
X_CLR_WR1	00000029	R	02
X_COM_WRO	*****	X	02
X_COM_WR1	0000002C	R	02
X_MOV_WRR	*****	X	02
X_SHIFT_R	*****	X	02
Y	= 00000044		
YBUSRD	= 000000EC		
ZERO_DAT	00000054	R	02
ZERO_SHIFT	0000005A	R	02

! Psect synopsis !													
PSECT name	Allocation		PSECT No.	Attributes									
. ABS .	00000000	(0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC
. BLANK .	00000000	(0.)	01 (1.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC
CPU2	00000DDE	(3550.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC

! Performance indicators !			
Phase	Page faults	CPU Time	Elapsed Time
Initialization	23	00:00:00.06	00:00:00.38
Command processing	34	00:00:00.23	00:00:01.14
Pass 1	1365	00:00:35.20	00:01:08.90
Symbol table sort	5	00:00:00.33	00:00:00.83
Pass 2	1415	00:00:13.27	00:00:27.15
Symbol table output	37	00:00:00.14	00:00:00.50
Psect synopsis output	5	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2889	00:00:49.26	00:01:38.93

The working set limit was 1000 pages.
389975 bytes (762 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 222 non-local and 117 local symbols.
4751 source lines were read in Pass 1, producing 52 object records in Pass 2.
151 pages of virtual memory were used to define 150 macros.

! Macro library statistics !	
Macro library name	Macros defined
SYSS\$SYSROOT:[SYSLIB]STARLET.MLB;1	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/SHOW=(BIN)/LIST=ENKBB.LIS/OBJECT=ENKBB.OBJ 8085MAC.MAR+MACDEF.MAC+[]ENKBB.MAC